

BITSTREAM Command User Manual

Overview

The BITSTREAM command generates extremely accurate bit sequences on one or two GPIO pins. Each timing value in the array specifies the duration (in microseconds) before the pin state toggles. This enables precise generation of protocols like IR remote control signals, PS/2 keyboard/mouse communication, and other timing-critical waveforms.

Syntax

Single Pin Mode

```
BITSTREAM pin, count, array()  
BITSTREAM pin, count, array(), mode
```

Dual Pin Mode

```
BITSTREAM pin1, count1, array1(), mode1, pin2, count2, array2()  
BITSTREAM pin1, count1, array1(), mode1, pin2, count2, array2(), mode2  
BITSTREAM pin1, count1, array1(), mode1, pin2, count2, array2(), mode2, logic
```

Parameters

Parameter	Description
`pin`, `pin1`, `pin2`	GPIO pin number(s) for the output. Can use GP notation (e.g., GP15) or physical pin numbers.
`count`, `count1`, `count2`	Number of timing values to use from the array (1 to 10000).
`array()`, `array1()`, `array2()`	Numeric array containing timing values in microseconds. Each value specifies the time before the next pin toggle.
`mode`, `mode1`, `mode2`	Output mode (optional, defaults to 0): **0** = Push-pull (driven high and low) **1** = Open-collector (driven low, high-impedance for high)
`logic`	Logic function for same-pin mode (optional, defaults to 0, only valid when pin1 = pin2): **0** = XOR (toggle on any event, original behavior) **1** = AND (output HIGH only when both logical states are HIGH) **2** = OR (output HIGH when either logical state is HIGH)

Output Modes

Push-Pull Mode (mode = 0)

The pin is actively driven both high and low. This is the default mode and is suitable for most applications including:

- IR LED driving (with appropriate current limiting)
- Direct digital outputs
- Single-ended signaling

Open-Collector Mode (mode = 1)

The pin is driven low for logic 0, and set to high-impedance (input with pull-up) for logic 1. This mode is required for bidirectional buses where multiple devices share the same line:

BITSTREAM Command User Manual

- PS/2 keyboard/mouse interfaces
- I2C-like protocols
- Any bus requiring wired-AND logic

Important considerations for open-collector mode:

1. Initial state: The pin is automatically initialized to high-impedance (HIGH) when the command is executed. Any prior pin state set by the user is overridden.
2. First transition is always LOW: Since the pin starts HIGH (hi-Z), the first toggle always drives the pin LOW.
3. Even number of transitions required: The count parameter must specify an even number of transitions. This ensures the pin returns to the HIGH (hi-Z) idle state after transmission. An error is generated if an odd count is specified.

Note: When using open-collector mode, ensure appropriate pull-up resistors are connected to the signal lines (typically 4.7kOhm to 10kOhm to VCC).

Dual Pin Operation

When two pins are specified, both bitstreams run simultaneously with their timing events merged chronologically. This enables:

1. Two Independent Signals: Generate clock and data lines for protocols like PS/2
2. Carrier Modulation: Use the same pin for both channels to modulate a carrier frequency with an envelope (e.g., IR protocols)

Same Pin for Both Channels

When pin1 equals pin2, the two channels control the same physical pin. The logic parameter determines how the logical states combine:

XOR Mode (logic = 0, default):

- The pin toggles on every event from either channel
- Original behavior for amplitude modulation where toggles interrupt the waveform

AND Mode (logic = 1):

- Each channel maintains a logical state (HIGH/LOW)
- Physical output is HIGH only when BOTH channels are logically HIGH
- Ideal for IR carrier modulation: carrier toggles fast, envelope enables/disables
- The carrier only appears on the output when the envelope is "open"

OR Mode (logic = 2):

- Physical output is HIGH when EITHER channel is logically HIGH
- Useful for combining two pulse trains

Example use case for AND mode (IR modulation):

- Channel 1: 38kHz carrier (rapid toggling between HIGH and LOW)
- Channel 2: Envelope (slow transitions: HIGH=carrier enabled, LOW=carrier disabled)
- Result: 38kHz bursts only during envelope HIGH periods

Timing Accuracy

BITSTREAM Command User Manual

- Timing values are specified in microseconds
- Maximum timing value: 67,108 microseconds (~67ms)
- Minimum recommended timing: ~1-2 microseconds (depends on CPU speed)
- Interrupts are disabled during transmission for maximum accuracy
- The first toggle occurs after the first timing value elapses

How It Works

1. The pin starts in its current state (typically low if just configured as output)
2. For each timing value in sequence:
 - Wait for the specified duration
 - Toggle the pin state
3. In dual-pin mode, events from both arrays are processed in chronological order
4. If events on different pins occur at the same time, both are processed

Examples

Example 1: Simple Square Wave

Generate a 1kHz square wave (500µs high, 500µs low):

```
Dim timing(100) As Integer
Local i As Integer
```

```
' Fill array with 500µs intervals
For i = 0 To 99
    timing(i) = 500
Next i
```

```
SetPin GP15, DOUT
Pin(GP15) = 0
BITSTREAM GP15, 100, timing()
```

Example 2: NEC IR Protocol with AND Logic

Generate a complete NEC IR remote control signal using AND logic for carrier gating:

```
' NEC IR Transmitter - Address 123, Command 107
' Uses AND logic: output = carrier AND envelope
' Carrier toggles at 38kHz; envelope gates carrier on/off
```

```
Const IR_PIN = 15
Const CARRIER_PERIOD = 13      ' microseconds (half period for 38kHz)
Const PULSE_562 = 562
Const SPACE_562 = 562
Const SPACE_1687 = 1687
Const LEADER_PULSE = 9000
Const LEADER_SPACE = 4500
Const ADDRESS = 123
Const COMMAND = 107
```

BITSTREAM Command User Manual

```
Dim envelope(100) As Integer
Dim envIdx As Integer = 0

' Leader: 9ms pulse, 4.5ms space
envelope(envIdx) = 0 : envIdx = envIdx + 1          ' t=0: enable carrier
envelope(envIdx) = LEADER_PULSE : envIdx = envIdx + 1 ' t=9000: disable for space
envelope(envIdx) = LEADER_SPACE : envIdx = envIdx + 1 ' t=13500: enable for bits

Sub AddBit(bitVal As Integer)
    Local space As Integer
    If bitVal = 0 Then space = SPACE_562 Else space = SPACE_1687
    envelope(envIdx) = PULSE_562 : envIdx = envIdx + 1 ' Disable after pulse
    envelope(envIdx) = space : envIdx = envIdx + 1     ' Enable after space
End Sub

' Send address, inverted address, command, inverted command (LSB first)
Dim i As Integer
For i = 0 To 7 : AddBit((ADDRESS >> i) And 1) : Next i
Dim invAddr As Integer = (Not ADDRESS) And &HFF
For i = 0 To 7 : AddBit((invAddr >> i) And 1) : Next i
For i = 0 To 7 : AddBit((COMMAND >> i) And 1) : Next i
Dim invCmd As Integer = (Not COMMAND) And &HFF
For i = 0 To 7 : AddBit((invCmd >> i) And 1) : Next i

' Final pulse
envelope(envIdx) = PULSE_562 : envIdx = envIdx + 1

' Calculate carrier transitions needed
Dim totalTime As Integer = 0
For i = 0 To envIdx - 1 : totalTime = totalTime + envelope(i) : Next i

Dim carrierCount As Integer = (totalTime \ CARRIER_PERIOD) + 50
Dim carrier(carrierCount) As Integer
carrier(0) = 0 ' First toggle at t=0
For i = 1 To carrierCount - 1 : carrier(i) = CARRIER_PERIOD : Next i

SetPin IR_PIN, DOUT
Pin(IR_PIN) = 0

' AND logic (1): carrier appears only when envelope is HIGH
BITSTREAM IR_PIN, carrierCount, carrier(), 0, IR_PIN, envIdx, envelope(), 0, 1
Print "NEC IR sent!"
```

Example 3: PS/2 Keyboard Scan Code

Send a PS/2 keyboard scan code using open-collector mode:

```
' PS/2 Keyboard Scan Code Transmitter
' Uses dual bitstream with open-collector mode

Const CLK_PIN = 15
Const DAT_PIN = 16
Const CLK_LOW = 40          ' Clock low time (µs)
Const CLK_HIGH = 40         ' Clock high time (µs)
Const INIT_DELAY = 50       ' Initial delay before first clock
```

BITSTREAM Command User Manual

```
' Transmit a PS/2 scan code
Sub PS2_Send(scanCode As Integer)
    Local bits(10) As Integer
    Local clkTiming(30) As Integer
    Local datTiming(30) As Integer
    Local clkIdx As Integer = 0
    Local datIdx As Integer = 0
    Local currentData As Integer = 1 ' Data starts HIGH (hi-Z)
    Local dataTime As Integer = 0
    Local bitVal As Integer
    Local i As Integer
    Local parity As Integer
    Local count As Integer = 0
```

```
' Calculate parity (odd parity)
For i = 0 To 7
    If (scanCode >> i) And 1 Then count = count + 1
Next i
parity = (count + 1) And 1
```

```
' Build the 11-bit frame: Start, D0-D7, Parity, Stop
bits(0) = 0 ' Start bit
For i = 0 To 7
    bits(i + 1) = (scanCode >> i) And 1
Next i
bits(9) = parity
bits(10) = 1 ' Stop bit
```

```
' First bit - data changes before first clock falling edge
bitVal = bits(0)
If bitVal <> currentData Then
    datTiming(datIdx) = INIT_DELAY - 5 : datIdx = datIdx + 1
    currentData = bitVal
    dataTime = 5
Else
    dataTime = INIT_DELAY
EndIf
```

```
' First clock transitions
clkTiming(clkIdx) = INIT_DELAY : clkIdx = clkIdx + 1
clkTiming(clkIdx) = CLK_LOW : clkIdx = clkIdx + 1
```

```
' Process remaining bits (1-10)
For i = 1 To 10
    bitVal = bits(i)
    If bitVal <> currentData Then
        datTiming(datIdx) = dataTime + CLK_LOW + 5 : datIdx = datIdx + 1
        currentData = bitVal
        dataTime = CLK_HIGH - 5
    Else
        dataTime = dataTime + CLK_LOW + CLK_HIGH
    EndIf
```

BITSTREAM Command User Manual

```
    clkTiming(clkIdx) = CLK_HIGH : clkIdx = clkIdx + 1
    clkTiming(clkIdx) = CLK_LOW : clkIdx = clkIdx + 1
Next i
```

```
' Return data to HIGH if needed (ensure even transitions)
If currentData = 0 Then
    datTiming(datIdx) = dataTime + CLK_LOW + CLK_HIGH : datIdx = datIdx + 1
EndIf
If (datIdx And 1) = 1 Then
    datTiming(datIdx) = 50 : datIdx = datIdx + 1
EndIf
```

```
' Transmit using open-collector mode
BITSTREAM CLK_PIN, clkIdx, clkTiming(), 1, DAT_PIN, datIdx, datTiming(), 1
End Sub
```

```
' Initialize and send scan code for 'A'
SetPin CLK_PIN, DOUT
SetPin DAT_PIN, DOUT
PS2_Send(&H1C)
Print "PS/2 scan code sent!"
```

Notes

- The pin must be configured as a digital output (DOUT) or unconfigured before use
- Interrupts are disabled during transmission to ensure timing accuracy
- For very long transmissions, be aware that other time-critical operations will be blocked
- In push-pull mode, the pin starts in whatever state you set it to before calling BITSTREAM
- In open-collector mode, the pin is forced to high-impedance (HIGH) at the start, and the count must be even to ensure it returns to HIGH
- Array indices start at 0; ensure the array has at least count elements

Error Messages

Error	Cause
Invalid pin	Pin number is out of range or refers to a reserved pin
Pin already in use	Pin is configured for another function
Array too small	Array has fewer elements than the specified count
Number range	Timing value exceeds 67,108 microseconds or is negative
Open-collector mode requires even number of transitions	Count must be even when using open-collector mode (mode = 1)
Logic parameter only valid when both pins are the same	The logic parameter (9th) can only be used when pin1 = pin2
Syntax error	Incorrect number of parameters (must be 3, 4, 7, 8, or 9)

BITSTREAM Command User Manual

See Also

- SETPIN - Configure pin function
- PIN() - Read or write digital pin state
- PULSE - Generate a single pulse
- PWM - Generate continuous PWM output