Programming the ATtiny10 [Updated]

11th November 2017

This article describes how to program the ATtiny10, Microchip's diminuitive 6-pin processor, using the Arduino IDE. It's a great chip for building small gadgets and wearables, or designing interface logic for other projects, and it really lives up to its "tiny" name:

The following sections explain how to program the ATtiny10 in C, and how to download programs using a low-cost ISP programmer. It also illustrates some simple applications with example programs.

Examples

For full projects based on the ATtiny10 see the following examples:

- ATtiny10 POV Pendant
- ATtiny10 Thermometer
- Morse Code Message Pendant
- Twinkling Pendant

Introduction

If, like me, you like using the simplest possible chip for each application, the ATtiny10 will appeal to you [1]; it's a 6-pin processor, about the same size as an 0805 SMD resistor, and it costs about 35p/35¢. It packs in the following features:

- Internal 8MHz clock, by default prescaled to 1MHz.
- Three I/O lines.
- Two 16-bit PWM analogue outputs.
- Three 8-bit analogue inputs.
- An analogue comparator.
- A 16-bit timer with input capture and an event counter.
- A watchdog timer.
- 1024 bytes of program memory, 32 bytes of RAM, and no EEPROM.

All of these features will be familiar to users of the larger AVR chips. Here's the pinout (using Spence Konde's design conventions):

Search

Most popular posts

Tiny Time 2 Watch
Minimal RP2040 Board
RA4M1 Nano Board
Programming the ATtiny10 [Updated]
Tiny Graphics Library

Recent posts

▼ 2024
Building Gates with the AVR Event System
RA4M1 Nano Board
Silver Dollar Game: How to Win
► 2023
► 2022
► 2021
► 2020
► 2019
► 2018
► 2017
► 2016
► 2015
► 2014

Topics

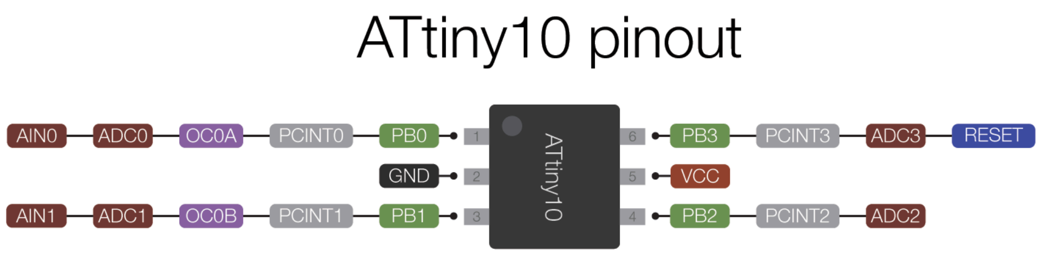
► Games
► Sound & Music
► Watches & Clocks
► GPS
► Power Supplies
► Computers
► Graphics
► Thermometers
► Wearables
► Test Equipment
► Tutorials
► Libraries
► PCB-Based Projects

By processor

AVR ATtiny

► ATtiny10
► ATtiny2313
► ATtiny84
► ATtiny841
► ATtiny85
► ATtiny861
► ATtiny88

AVR ATmega



The internal oscillator is accurate to within 10%, but you can calibrate it in software to within 1%. You can configure RESET as a fourth I/O line, which prevents further programming, but I don't cover that in this article.

To work with the ATtiny10 on a breadboard you can mount it on a SOT23 breakout board, such as the one available from Sparkfun [2].

Programming the ATtiny10

Unlike the SPI protocol used to program the larger AVR chips, such as the ATmega328 in the Arduino Uno, the ATtiny10 uses a programming protocol called TPI (Tiny Programming Interface) which needs only five wires. Fortunately Thomas Fischl's excellent USBasp programmer supports this protocol [3]; you can build your own, order one from his site, or they are widely available on eBay [4], Banggood [5], etc. I recommend getting one with a 10-pin to 6-pin adapter for ISP programming. The current versions of the Arduino IDE support the ATtiny10, so you can program it in C and upload programs as easily as with the other AVR chips. Since an Arduino core would use up almost half of the available program memory the best way to program it is to access the registers directly, and I give an overview of how to do this in the section Alternatives to core functions below.

Here are instructions for programming the ATtiny10.

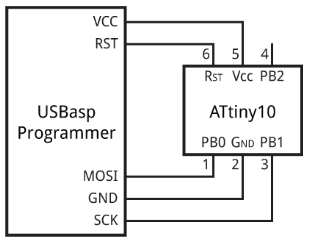
- Install the **ATtiny10Core** by following the instructions on my GitHub repository ATtiny10Core.

This should add an **ATtiny10Core** heading to the **Board** menu.

- Enter your program into the Arduino IDE editor.

For example, try the **Blink** example program given below.

- Connect the USBasp to the ATtiny10 as shown in the following diagram:



Connecting the USBasp programmer to an ATtiny10.

- Choose **Board** from the **Tools** menu, and select the **ATtiny10/9/5/4** option under the **ATtiny10Core** heading; it's the only option.
- Choose the chip you want from the **Chip** menu; for example **ATtiny10**.
- Choose **USBasp** from the **Programmer** option on the **Tools** menu.
- Choose **Upload** to upload the program.

The LED should blink at 0.5Hz.

Here's my test setup on a mini breadboard:

► ATmega328
► ATmega1284

AVR 0-series and 1-series

► ATmega4809
► ATtiny1604
► ATtiny1614
► ATtiny3216
► ATtiny3227
► ATtiny402
► ATtiny404
► ATtiny414
► ATtiny814

AVR DA/DB-series

► AVR128DA28
► AVR128DA32
► AVR128DA48
► AVR128DB28

ARM

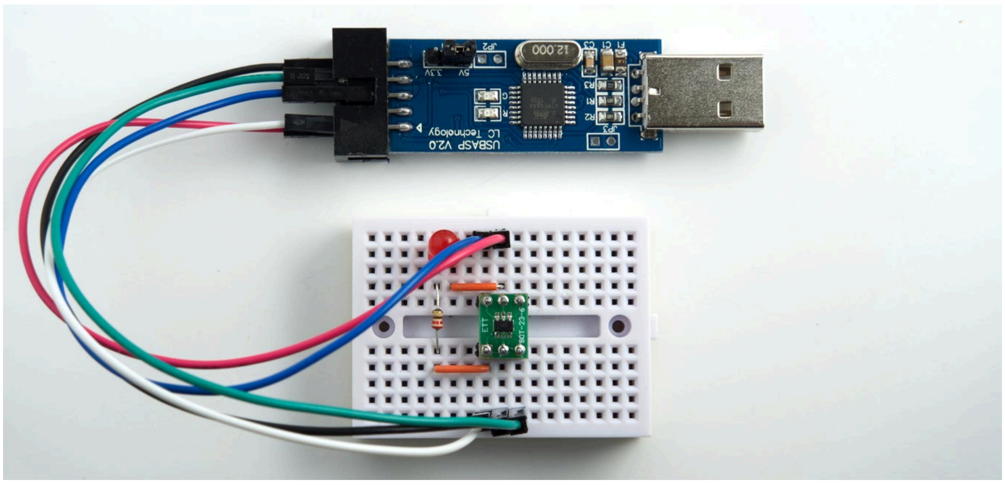
► ATSAMD21
► RP2040
► RA4M1

About me

About me
Contact me
Twitter
Mastodon

Feeds

RSS feed



Testing the ATtiny10 Blink program on a mini breadboard, using the USBasp programmer.

Examples

Here are a couple of examples using the ATtiny10:

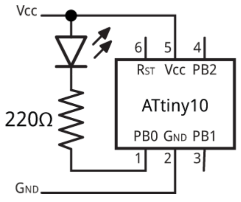
Blink

This is the ubiquitous Blink program:

```
void setup () {
  DDRB = 1;                         // PB0 as an output
  TCCR0A = 1<<COM0A0 | 0<<WGM00;  // Toggle OC0A, CTC mode
  TCCR0B = 1<<WGM02 | 3<<CS00;    // CTC mode; use OCR0A; /64
  OCR0A = 15624;                    // 1 second; ie 0.5Hz
}

void loop () {
}
```

To run it connect an LED to PB0 as follows:



Circuit using an ATtiny10 to blink an LED.

It uses Timer/Counter0 to divide the 1MHz system clock by a prescaler value of 64, and then by 15625, toggling the output PB0 with a period of 1 second.

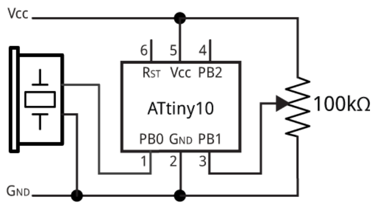
Analogue frequency generator

The following program reads the voltage from a potentiometer on analogue input ADC1 (PB1), and then uses this to set the compare match register OCR0A of Timer/Counter0, to generate a square wave on PB0 whose frequency you can control with the potentiometer:

```
void setup () {
  DDRB = 1;                         // PB0 as an output
  // Set up ADC on PB2
  ADMUX = 1<<MUX0;                  // ADC1 (PB1)
  ADCSRA = 1<<ADEN | 3<<ADPS0;     // Enable ADC, 125kHz clock
  // Set up waveform on PB0
  TCCR0A = 1<<COM0A0 | 3<<WGM00;  // Toggle OC0A, Fast PWM
  TCCR0B = 3<<WGM02 | 4<<CS00;    // Fast PWM with OCR0A as TOP; /256
}

void loop () {
  ADCSRA = ADCSRA | 1<<ADSC;      // Start
  while (ADCSRA & 1<<ADSC);       // Wait while conversion in progress
  OCR0A = ADCL;                   // Copy result to frequency output
}
```

Note that because we're changing the compare match value, we need to use Fast PWM mode in this application, because it double-buffers the compare match value. Here's the circuit:



Circuit using a potentiometer to adjust the frequency of a square wave generated by an ATtiny10.

It generates a frequency between 1MHz/256/256, or about 15Hz, and 1MHz/256/1, or 3.9kHz.

Alternatives to core functions

The following sections give some tips on programming the ATtiny10 to achieve some of the things provided by the Arduino core functions.

pinMode

To specify whether pins are inputs or outputs you set the corresponding bits in the **DDRB** register to 0 or 1 respectively. For example, to define pins 1 and 3 as outputs (and leave the other pins as inputs):

```
DDRB = 0b0101;         // Equivalent to pinMode(1, OUTPUT); pinMode(3, OUTPUT);
```

Input pullups

Unlike the older AVR chips, such as the ATmega328 and ATtiny85, the ATtiny10 enables pullup resistors using a separate pullup register, **PUEB**. To set pullups on input pins you set the corresponding bits in this register. For example, to set a pullup resistor on input pin 2:

```
PUEB = 0b0010;         // Equivalent to pinMode(2, INPUT_PULLUP);
```

Note that it doesn't make sense to set a pullup on an output.

digitalWrite

To set the state of an output you set the corresponding bits in the **PORTB** register. For example, to set bit 1 low and bit 3 high (assuming they have been defined as outputs):

```
PORTB = 0b0100;        // Equivalent to digitalWrite(1, LOW); digitalWrite(3, HIGH);
```

Changing the state of an input has no effect.

digitalRead

To read the state of the I/O pins you read the **PINB** register:

```
int temp = PINB;
```

analogWrite

You can use OC0A (PB0) and OC0B (PB1) for analogue output using PWM. You first need to configure the Timer/Counter into PWM mode for that pin; for example, using PB0:

```
TCCR0A = 2<<COM0A0 | 3<<WGM00; // 10-bit PWM on OC0A (PB0), non-inverting mode
TCCR0B = 0<<WGM02 | 1<<CS00;   // Divide clock by 1
DDRB = 0b0001;                 // Make PB0 an output
```

To write an analogue value we then need to write the value to the appropriate output compare register, OCR0A:

```
OCR0A = 1000;                  // Equivalent to analogWrite(0, 1000)
```

With a 5V supply this will set PB0 to 1000/1024 * 5V, or 4.88V.

analogRead

To use an I/O pin for analogue input you first need to configure the Analogue-to-Digital Converter. For example, to use ADC0:

```
ADMUX = 0<<MUX0;               // ADC0 (PB0)
ADCSRA = 1<<ADEN | 3<<ADPS0;   // Enable ADC, 125kHz clock
```

To read an analogue value from the pin we then need to start a conversion, and when the conversion is ready read the ADC register:

```
ADCSRA = ADCSRA | 1<<ADSC;     // Start
while (ADCSRA & 1<<ADSC);      // Wait while conversion in progress
int temp = ADCL;               // Copy result to temp
```

delay

For a simple substitute for **delay()** you can use a loop adjusted to give roughly the right timing:

```
void delay (int millis) {
  for (volatile unsigned int i = 34*millis; i>0; i--);
}
```

This would provide an alternative way of writing the Blink program. Note that the counter variable **i** must be defined as **volatile** or the compiler will optimise it out of the loop, eliminating the delay.

For more accurate delays, and to implement timers like **millis()**, you could set up Timer/Counter0 as a timer, or use the Watchdog Timer.

Updates

6th March 2021: Updated the description to refer to the new ATtiny10Core and Boards Manager installation, designed to solve problems using the ATtiny10Core on recent versions of the Arduino IDE. I've removed comments relating to this from the discussion below to avoid confusion.

11th January 2022: I've updated the link to the ATtiny10 datasheet to point to the latest 2018 version. The previous 2016 version described the operation of I/O port pullups incorrectly.

---

1. ^ ATtiny10 Datasheet on Microchip.
2. ^ Sparkfun SOT23 to DIP Adapter on Sparkfun.
3. ^ USBasp - USP programmer for Atmel AVE controllers on www.fischl.de.
4. ^ USBASP ISP Programmer Cable Adapter from Boos Bits on eBay.
5. ^ USBASP 3.3 5V AVR Downloader Programmer on Banggood.

Next: Continuity Tester **Previous:** Proto Power Supply

36 Comments 1 Login ▾

G Join the discussion…

LOG IN WITH OR SIGN UP WITH DISQUS ?

Name

♡ 6 Share Best Newest Oldest

Joe
2 years ago

Well, add me to the frustrated. I update 3 usbasp's and still cannot upload thru programmer. I also tried to place 12v instead on the reset. Also tried jumpering the slow clock.

avrdude: error: wrong responds size
avrdude: error: program enable: target doesn't answer.
avrdude: initialization failed, rc=-1
Double check connections and try again, or use -F to override
this check.

I would really like to know why this doesn't work.

1 0 Reply

johnsondavies Mod → Joe
2 years ago

Have you connected it up exactly as in my photograph?

0 0 Reply

Joe → johnsondavies
2 years ago edited

yep. I have an attiny10 breakout board that has a cap on vcc and a switch to gnd on pb3.(not connected when sw not pressed), the pins go to: vcc, gnd, pb0,pb1,pb2, and pb3,. Wired to the usbasp, nothing else is connected to the chip. The chip will program with the procedure here: http://junkplusarduino.blog... with an arduino, but not with a usbasp. I have one with an atmega8 and two with atmega48, updated both to the latest revision, and it still fails. I'm using Windows 10 and the libusb-win32 driver. (this works with usbasp and all other devices).

0 1 Reply

Giselle → Joe
2 years ago edited

You have the jumper set to 5V and not 3V? It will not program unless it is at 5V. Did you select the USBasp as your programmer? **Did you set the programming clock speed to 1MHz on the USBasp? Sometimes when you buy the USBasp, they do not solder any pins into that connector. It's J3 or JP3. If you don't short that, it may be trying to program at high speed but these chips program at low speed.**

Are you using the procedure through Arduino IDE or interacting with avrdude directly? I ask because you mentioned your driver..But I think you don't need to worry about the driver if you're using the Arduino IDE, it's already included.

Bolded the most common problem.

0 0 Reply

Joe → Giselle
2 years ago edited

Yep to all. jumpered for 5v to target and j3 jumpered. Waiting on some

new chips, will try when they come. Then, tried on an uno and it didn't like it. Redid the driver for win-usb and finally! programmed the attiny10. I could've swore I tried a different driver before.
Edit: Found with the driver that works, only the USBasp with the ATMEG8 works, the other two have a problem with Arduino IDE. This is really strainge since I have used these many times before.

0 0 Reply

A

Arnab Ghosh ➦ Joe
2 years ago

Face the same issue.
Then I update driver with **Zadig** to **WinUSB (v6.1.7600.16385)**.
Now also showing "An error occurred while uploading the sketch"

But actually uploaded and working.

0 0 Reply

YB

Yago Bueno ➦ Arnab Ghosh
2 years ago edited

2 year's ago I used my arduino nano as programmer for attiny 10.
I found a site who describe very well about how to do this, including a .ino and java program.

> `http://junkplusarduino.blogspot.com/p/attiny10-resources.html`

0 0 Reply

I

Isaac
4 years ago edited

Quick question (I think I've revisited this post once a month for the last two years!): I have a chunk of Attiny10 code that runs on Pin-Change Interrupt on PB0 (PCINT0). I'd like to add a push and hold type action, but I can't quite get it working. I'm able to detect falling/rising without issue though. Any idea what I would need to add?

Ideally, I'm trying for:
Action #1. User presses button once, code runs as normal
Action #2. User presses and holds button for ~ 3 seconds, a special function executes

```
ISR(PCINT0_vect) {
if(!pendingChange && PINB & (1 << PB0)) {
// Button has been pressed!
pendingChange = true;

} else {
// Button has been released!

}
}
```

1 0 Reply

I

Isaac ➦ Isaac
4 years ago edited

Ah Ha! I figured it out:

```
int main(void) {
while(1) {
while((PINB & (1 << PB0))) {
buttonHeldDown++;
_delay_ms(10);
if (buttonHeldDown >= 200 && !doingStuff) {
// Awesome! Button was held down for ~ 2 seconds
}
}
```

0 0 Reply

johnsondavies Mod ➜ Isaac
4 years ago

Great! Glad you solved it.

0 0 Reply

Joe
2 years ago

Ok, after all the discussion, got an ATTiny10 and soldered to a breakout board and programmed a slower Blink program on it. It is now an aircraft warning on top of my N scale water tower.

0 0 Reply

johnsondavies Mod ➜ Joe
2 years ago

Good to hear!

0 0 Reply

Sullivan Z
2 years ago

How to implement low power sleep on ATtiny10?
I had some experience with ATtiny24/44/84 low power sleep with watchdog timer set up to wake up periodically. I read datasheet for ATtiny10 but I am puzzled that on ATtiny10, you need to execute SLEEP asm command explicitly to get ATtiny10 to start sleeping. Do you have any experience?

0 0 Reply

johnsondavies Mod ➜ Sullivan Z
2 years ago

See my latest Morse Code Message Pendant project that does exactly what you want to do.

1 0 Reply

Sullivan Z ➜ johnsondavies
2 years ago

That's really cool :)))) Many thanks.

1 0 Reply

taylor
4 years ago

Hi and huge thank you for this tutorial. I'm working with usbasp programmer + avrdude and am wondering why I cannot program the chip with any components connected to it. I noticed in a comment below there was a suggestion to make sure all components are disconnected, or they may interfere w/ TPI, but now I'd like to make some PCBs and not sure how to do this. Any suggestions on where to look? thank you!

0 0 Reply

johnsondavies Mod ➜ taylor
4 years ago

I suggest you test it on a breadboard first. Sometimes putting a resistor in series with the pin helps isolate the external components from interfering with programming. If all else fails you could make a jig to program the ATtiny10 before you solder it on the PCB.

0 0 Reply

taylor ➜ johnsondavies
4 years ago

Thanks @johnsondavies. I tried 10ohm to 10k series resistors on pin 0 and still couldn't upload to board. the main problem has been on pin 0. So if I can avoid connecting to that it should be ok, but kind of annoying. I think I'll end up making a jig... :/

0 0 Reply

Per Thomas Jahr

4 years ago

I also had problems with my USBasp, but I had great success with flashing the firmware of one of my USBasps by following the instructions given here:
https://blog.podkalicki.com...

After that I compiled and programmed the ATtiny10 by using the Makefile from the book "Make: AVR Programming" (needs som editing - see comments in file):
https://github.com/hexagon5...

With that Makefile in place one can just do "make flash" and code is compiled, linked and uploaded to target.

0 0 Reply

Brian M.
5 years ago

Hi! Great blog!
So happy that I finally read through and got this up and working.

One thing I'm struggling to understand though:
Why does **DDRB = 1;** set PB0 to output?
I would have thought you'd need to write **DDRB = 0b1000;**. Am I missing something obvious?

0 0 Reply

johnsondavies Mod → Brian M.
5 years ago edited

If you look at the datasheet (page 76) PB0 is the bottom bit in DDRB, and you need to set it to a '1' for an output, so the code is:

```
DDRB = 0b0001
```

which is the same as **DDRB = 1**.

0 0 Reply

Brian M. → johnsondavies
5 years ago

Ah, forgot it reads right to left as well!
This is perfect, thank you! :)

0 0 Reply

Isaac
5 years ago

Hey, really appreciate your work with this. I'm having the same issue that a few others are having where it says "Target Doesn't Answer". I think my jumper is on the default J1 setting. Is this correct option for flashing the attiny?

0 0 Reply

Stefan Huus Gregersen → Isaac
5 years ago

Hello mate! It is important that you update your USBASP to the newest firmware as the USBASP from china typically isn't updated to that
Also ensure that your driver is libusbK

1 0 Reply

Stefan Huus Gregersen
5 years ago

Man, been trying out to make this work but it just won't work. I am using the exact same USBASP as you, same wiring which i have ensured, also tried different USB ports.
Been stuck at this error for days
warning: cannot set sck period. please check for usbasp firmware update.
avrdude: error: program enable: target doesn't answer. 1
avrdude: initialization failed, rc=-1

Any idea why? I also just tried out Dayne's recommendations to apply 12V to reset, it didn't work..

0 0 Reply

johnsondavies Mod ➔ Stefan Huus Gregersen
5 years ago

Check that your USBasp has the latest firmware - see the link to www.fischl.de in the article. Only the latest firmware supports TPI.

0 0 Reply

Stefan Huus Gregersen ➔ johnsondavies
5 years ago

Thanks
Updated the firmware
Now i get this every time
I have ensured wiring is correct
(10x) avrdude: error: wrong responds size
avrdude: error: program enable: target doesn't answer.
avrdude: initialization failed, rc=-1
Double check connections and try again, or use -F to override
this check.

0 0 Reply

johnsondavies Mod ➔ Stefan Huus Gregersen
5 years ago

Try connecting USBasp to a bare ATtiny10, with no other components in the circuit. The other components may be interfering with the TPI.

0 0 Reply

Stefan Huus Gregersen ➔ johnsondavies
5 years ago

Hello
I am running just the IC and a 100n decoupling capacitor

0 0 Reply

johnsondavies Mod ➔ Stefan Huus Gregersen
5 years ago

I'm sorry, I'm not sure what to suggest. Are you sure your USBasp is working? Check you can program another chip such as an ATtiny85 with the USBasp. Try another ATtiny10 in case the first one has been damaged.

0 0 Reply

Stefan Huus Gregersen ➔ johnsondavies
5 years ago

I have used it before for programming different chips so it should work.
It just seems weird why it won't work

0 0 Reply

Dayne Waterlow
5 years ago edited

Well, I've been playing with this for a couple days, this little chips a pushing me to learn past the Arduino environment, your site is a lot of help!

Using your blink example, I would like to make it slower, but I'm not quite understanding it. I've got the slowest speed down by changing to:"

```
TCCR0B = 3<<WGM02 | 5<<CS00;
```

But how can I lower the maximum speed?

0 0 Reply

johnsondavies Mod ➔ Dayne Waterlow
5 years ago

I'm not quite sure what you mean by "slowest speed" and "maximum speed". The number in front of CS00 determines the scale factor, which gives a coarse control. My original value of 3 gives divide by 64. Your value of 5 gives divide by 1024, which will be 16 times slower. The value of OCR0A determines the divisor, which gives fine control. The maximum value you can put here is 65535 which would be a factor of about 4 slower than my value of 15624. You don't want to change the value in front of WGM02 - that's a different mode.

0 0 Reply

D

Dayne

5 years ago

I had the very same problem with most of my attiny10's from china. I read the data sheet and found the "high voltage programing" part. Basically, disconnect the RST pin from your USBASP and connect 12v to the RST pin of the tiny10, then program the chip and remove the 12v when its done. It should work after that.

0 0 Reply

Eric Navarrete

5 years ago

First of all, thanks for this great post. I have one question though, I am trying to read the state of PB2 to a variable, but I cant figure out the correct syntax to use. I thought it would be something like this:
int buttonState = PINB & 2;
but it doesn't seem to work
Any help is greatly appreciated

0 0 Reply

johnsondavies Mod → Eric Navarrete

5 years ago

I think you need:

```
int buttonState = PINB & 4;
```

since PB2 is bit 2, and 0b100 is 4.

0 0 Reply

Subscribe Privacy Do Not Sell My Data