# IsoMax Performance Monitoring

The IsoMax system is designed to execute user-defined state machines at a regular interval. This interval can be adjusted by the user with the `PERIOD` command. But how quickly can the state machine be executed? IsoMax provides tools to measure this, and also to handle the occasions when the state machine takes "too long" to process.

## *An Example State Machine*

For the purposes of illustration, we'll use a state machine that blinks the green LED:[1]

```
LOOPINDEX CYCLE-COUNTER
DECIMAL 100 CYCLE-COUNTER END
1 CYCLE-COUNTER START

MACHINE SLOW_GRN

ON-MACHINE SLOW_GRN
 APPEND-STATE SG_ON
 APPEND-STATE SG_OFF

IN-STATE SG_ON
 CONDITION CYCLE-COUNTER COUNT
 CAUSES GRNLED OFF
 THEN-STATE SG_OFF
 TO-HAPPEN

IN-STATE SG_OFF
 CONDITION CYCLE-COUNTER COUNT
 CAUSES GRNLED ON
 THEN-STATE SG_ON
 TO-HAPPEN

SG_ON SET-STATE
INSTALL SLOW_GRN
```

This machine will execute at the default rate of `DECIMAL 50000 PERIOD`, or 100 Hz (since the clock rate is 5 MHz).

## *IsoMax Processing Time*

Every time IsoMax processes your state machines, it measures the total number number of clock cycles required. This is available to you in three variables:

`TCFAVG` This is a moving average of the measured processing time.[2] It is reported as a number of 5 MHz clock cycles.

`TCFMIN` This is the minimum measured processing time (in 5 MHz cycles). Note that this is *not* automatically reset when you install new state machines. Therefore, after installing new state machines, store a large value in `TCFMIN` to remove the old (false) minimum.

---

[1] This example uses `LOOPINDEX` and `INSTALL`, and therefore requires IsoMax v0.36 or later.

[2] To be precise, `TCFAVG` is computed as the arithmetic mean of the latest measurement and the previous average, i.e., Tavg[n+1] = (Tmeasured + Tavg[n]) / 2.

TCFMAX This is the maximum measured processing time (in 5 MHz cycles). This is *not* automatically reset when you change state machines. Therefore, after changing state machines, store a zero in TCFMAX to remove the old (false) maximum.

To see this, enter the following commands while the SLOW_GRN state machine is running:

```
DECIMAL 50000 TCFMIN !
0 TCFMAX !
TCFAVG ?
TCFMIN ?
TCFMAX ?
```

You may see an AVG and MIN time of about 630 cycles, and a MAX time near 1175 cycles.[3] With a 5 MHz clock, this corresponds to a processing time of about 126 usec (average) and 235 usec (maximum). The average is near the minimum because most of the time, the state machine is performing no action. Only once every 100 iterations does the CYCLE-COUNTER expire and force a change of LED state.

TCFAVG, TCFMIN, and TCFMAX return results in the same units used by PERIOD (counts of a 5 MHz clock). This means you can use TCFMAX to determine the safe lower bound of PERIOD. In this case, you could set PERIOD as low as 1175 decimal, and IsoMax would always have time to process the state machine.

## *Exceeding the Allotted Time*

What if, in this example, PERIOD had been set to 1000 decimal? Most of the time, the state machine would be processed in less time, but once per second the LED transition would require more time than was allotted.

IsoMax will handle this gracefully by "skipping" clock interrupts as long as the state machine is still processing. With PERIOD set to 1000, an interrupt occurs every 200 usec. When the LED transition occurs, one interrupt will be skipped, and so there will be 400 usec (2000 cycles) between iterations of the state machine.

If this happens only rarely, it may not be of concern. But if it happens frequently, you may have a problem with your state machine, or you may have set PERIOD too low. To let you know when this is happening, IsoMax maintains an "overflow" counter:

TCFOVFLO A variable, reset to zero when IsoMax is started, and incremented every time a clock interrupt occurs before IsoMax has completed state processing. (In other words, this tells you the number of "skipped" clock interrupts.)

You can see this in action by typing the following commands while the SLOW_GRN state machine is still running:

```
TCFOVFLO ?
DECIMAL 1000 PERIOD
TCFOVFLO ?
TCFOVFLO ?
TCFOVFLO ?
50000 PERIOD
TCFOVFLO ?
TCFOVFLO ?
```

[3] These times were measured on an IsoPod running the v0.37 kernel. With no state machines INSTALLed, the same kernel shows a TCFAVG of 88 cycles (17.6 usec). This represents the overhead to respond to a timer interrupt, service it, and perform an empty INSTALL list.

Be sure to type these commands, and don't just upload them -- you need some time to elapse between commands so that you can see the overflow counter increase. After you change PERIOD back to 50000, the overflow counter will stop increasing.

## Automatic Overflow Processing

If IsoMax overflows happen too frequently, you may wish your application to take some corrective action. You could write a program to monitor the value of TCFOVFLO. But IsoMax does this for you, and allows you to set an "alarm" value and an action to be performed:

TCFALARM — A variable, set to zero when IsoMax is started. If set to a nonzero value, IsoMax will declare an "alarm" condition when the number of timer overflows (TCFOVFLO) reaches this value. If set to zero, timer overflows will be counted but otherwise ignored.

TCFALARMVECTOR — A variable, set to zero when IsoMax is started. If set to a nonzero value, IsoMax will assume that this is the CFA of a Forth word to be executed when an "alarm" condition is declared. This Forth word should be stack-neutral, that is, it should consume no values from the stack, and should leave no values on the stack.

If set to zero, timer overflows will be counted but otherwise ignored.

Note that *both* of these values must be nonzero in order for alarm processing to take place. Be particularly careful that TCFALARMVECTOR is set to a valid address; if it is set to an invalid address it is likely to halt the IsoPod.

To continue with the previous example:

```
REDLED OFF
: TOO-FAST   REDLED ON  50000 PERIOD ;
' TOO-FAST CFA  TCFALARMVECTOR !
100 TCFALARM !
0 TCFOVFLO !
```

This defines a word TOO-FAST which is to be performed if too many overflows occur. TOO-FAST will turn on the red LED, and will also change the IsoMax period to a large (and presumably safe) value. The phrase ' TOO-FAST CFA returns the Forth CFA of the TOO-FAST word; this can be stored as the TCFALARMVECTOR. Finally, the alarm threshold is set to 100 overflows, and the overflow counter is reset.[4]

Now watch the LEDs after you type the command

```
1000 PERIOD
```

The slow blinking of the green LED will change to a rapid flicker for a few seconds. Then the red LED will come on and the green LED will return to a slow blink. This was caused by the word TOO-FAST being executed automatically when TCFOVFLO reached 100.

## Counting IsoMax Iterations

It may be necessary for you to know how many times IsoMax has processed the state machine. IsoMax provides another variable to help you determine this:

TCFTICKS — A variable, set to zero when IsoMax is started, and incremented on every IsoMax clock interrupt.

---

[4] The test is for equality (TCFOVFLO=TCFALARM), not "greater than," to ensure that the alarm condition only happens once. The previous exercise left a large value in TCFOVFLO; if this is not reset to zero, the alarm won't occur until TCFOVFLO reaches 65535, "wraps around" back to zero, and then counts to 100.

The frequency of the IsoMax clock interrupt is set by `PERIOD`; the default value is 100 Hz (50000 cycles of a 5 MHz clock). With this knowledge, you can use `TCFTICKS` for time measurement. With `DECIMAL 50000 PERIOD`, the variable `TCFTICKS` will be incremented 100 times per second.

Note that `TCFTICKS` is incremented *whether or not* an IsoMax overflow occurs. That is, it counts the number of IsoMax clock interrupts, *not* the number of times the state machine was processed. To compute the actual number of executions of the state machine, you must subtract the number of "skipped" clock interrupts, thus:

```
TCFTICKS @ TCFOVFLO @ -
```