# Qspice - How Time Step Works

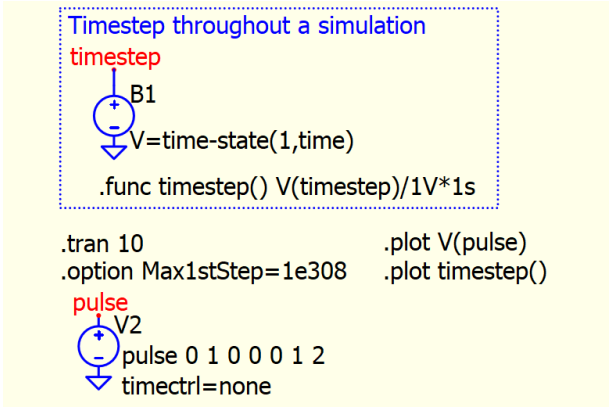
KSKelvin Kelvin Leung

Created on : 5-23-2024
Last Update : 9-30-2024

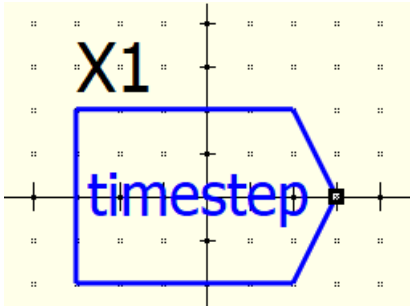
# How Time Step Works in Qspice – TimeStep

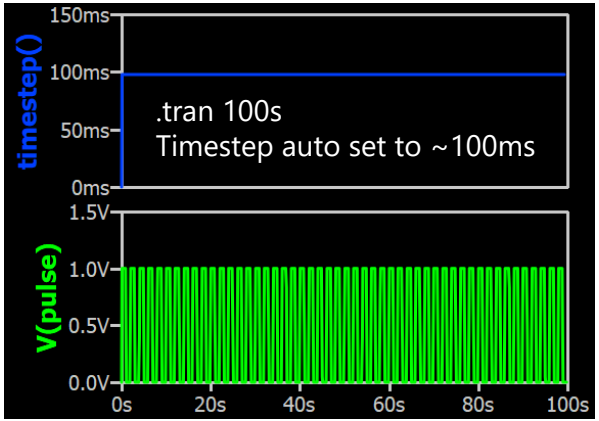
## Qspice : timestep.qsch

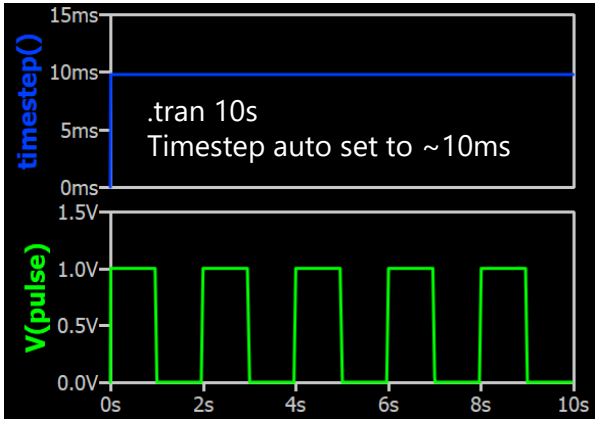
- Timestep
  - Simulation Time of Qspice can be calculated with the help of function state(n,x)
  - B-source with formula time-state(1,time)
    - Current time – value of time 1 time step ago
  - In KSKelvin's Symbol library, symbol Timestep.qsym is created to return timestep




Timestep.qsym



# How timestep works?

## Qspice : timestep- MaxStep.qsch | timestep - Pulse Timectrl.qsch

- #1a .option maxstep
  - Maximum timestep
- #1b .tran Tstart to Tstop
  - Without timestep modification devices, Qspice set a constant timestep
  - Timestep= $\min\left(\sim\frac{\text{Tstop}-\text{Tstart}}{1000}, \text{maxstep}\right)$
- #2a Timectrl Devices
  - Device (Voltage Source, Switch, ¥-Device etc…) can affect timestep
  - A voltage source with instance parameter Timectrl can reduce the timestep at signal break point

Timestep throughout a simulation

X1

timestep — timestep

.func timestep() V(timestep)/1V*1s

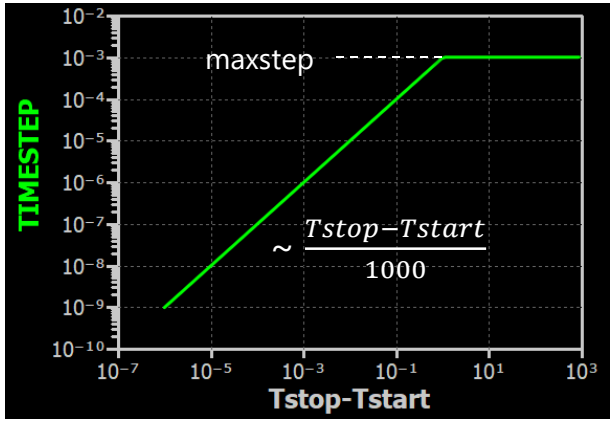
.tran 0 Tstop Tstart
.plot V(pulse)
.plot timestep()

pulse

V2 Timectrl=none

pulse 0 1 0 0 0 1 2

.param Tstart 1m
.step dec param Tstop 1m+1µ 1K 50
.meas timestep max V(timestep)
.option maxstep=1e-3




Timestep throughout a simulation

X1

timestep — timestep

.func timestep() V(timestep)/1V*1s

.tran 0 Tstop Tstart
.plot V(pulse)
.plot timestep()

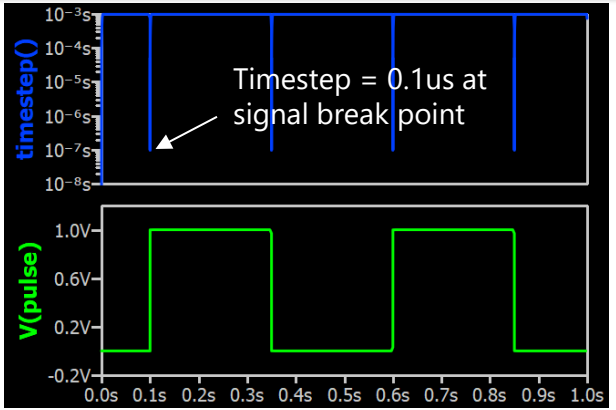
.param Tstart 0
.param Tstop 1
.option Max1stStep=1e308

pulse

V2 Enable Timectrl

Timectrl=none

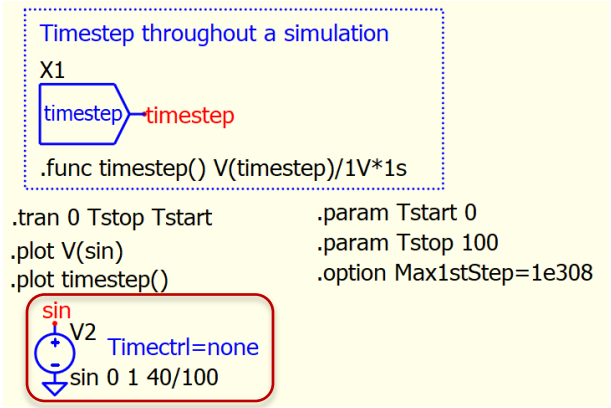
pulse 0 1 0.1 0 0 0.25 0.5

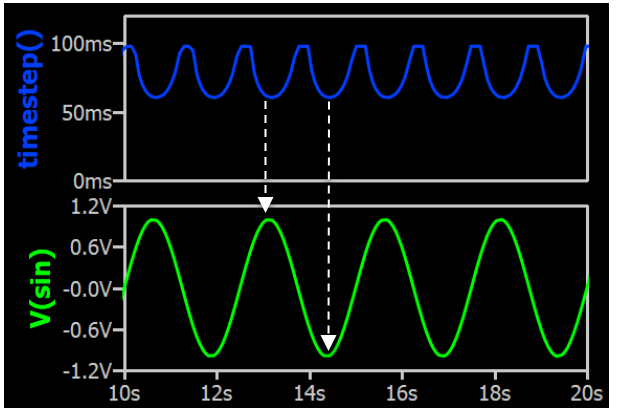


# How timestep works?

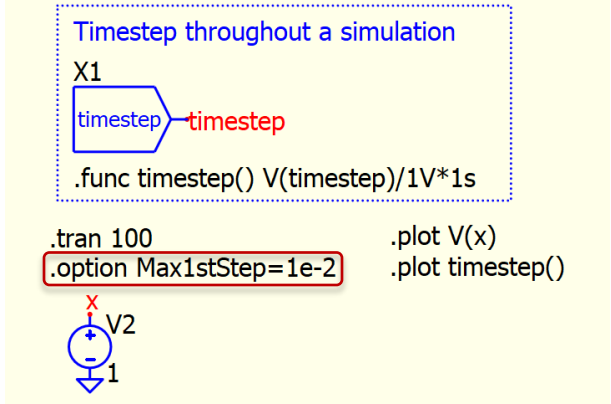
## Qspice : timestep - Sin Timectrl.qsch | timestep - Max1stStep.qsch

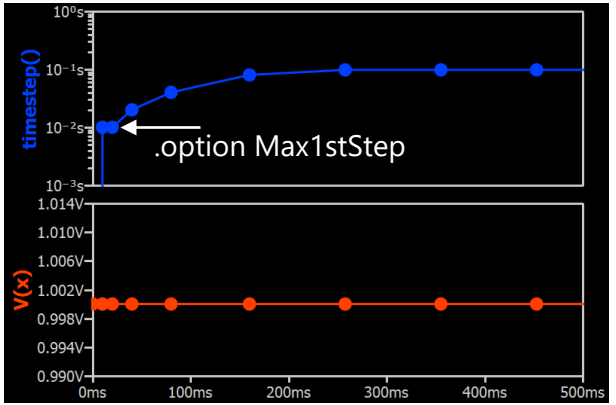
- #2b Timectrl Devices
  - V/I sources have different Timectrl strategies
    - For example, sine source reduce timestep when $\frac{dv}{dt}$ change direction
  - Setting the Instance parameter Timectrl=none for source will disable the timestep control strategy




- #3 .option Max1stStep
  - **.option Max1stStep** controls the maximum timestep size for the first timestep in a .tran
    - Default Max1stStep=100ns
  - To disable Max1stStep, set .option Max1stStep=1e308



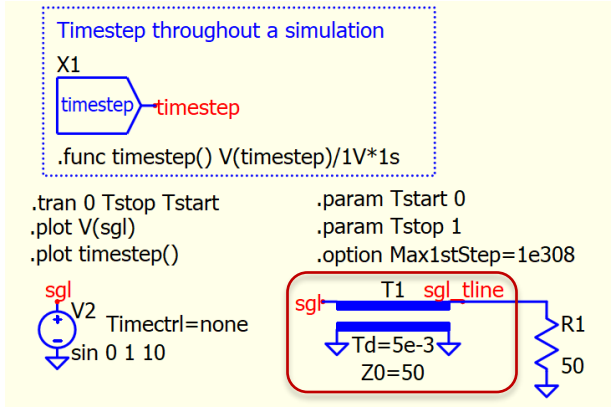



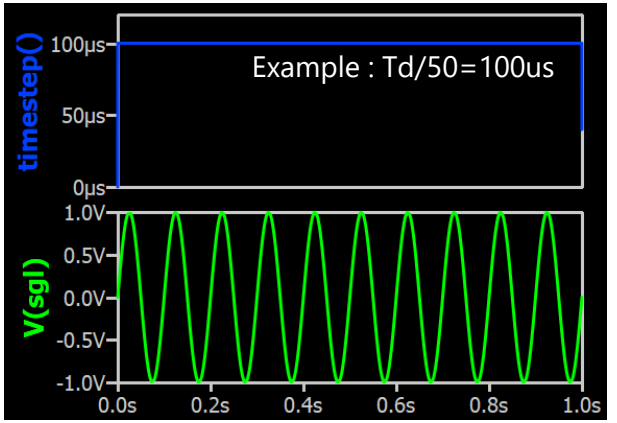
# How timestep works?

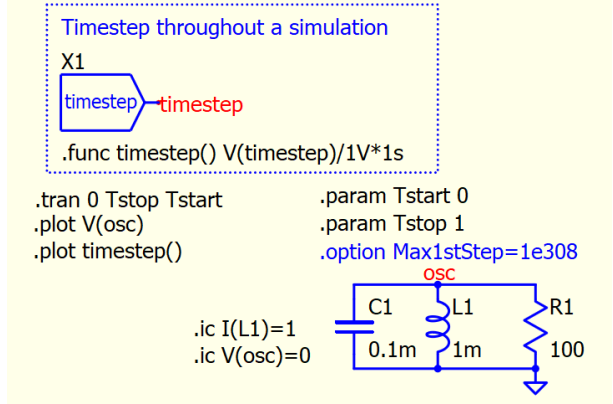
## Qspice : timestep - Tline.qsch | timestep - LC.qsch

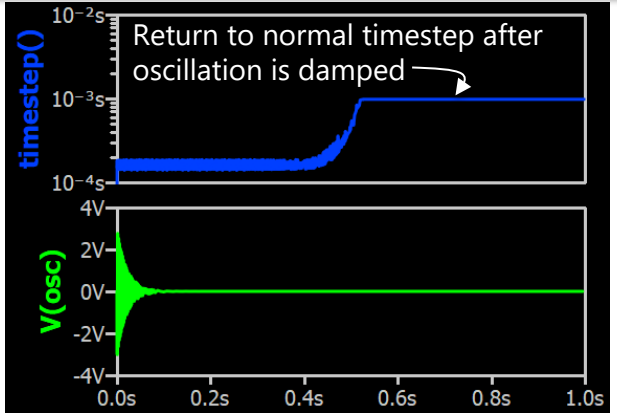
- #4 Transmission Line
  - Td of an ideal transmission line will force the target timestep to $\text{Timestep} = \frac{\text{Td}}{50}$

- #5 LC oscillation
  - Qspice can changes its timestep if circuit consist of resonant elements and before oscillation is damped



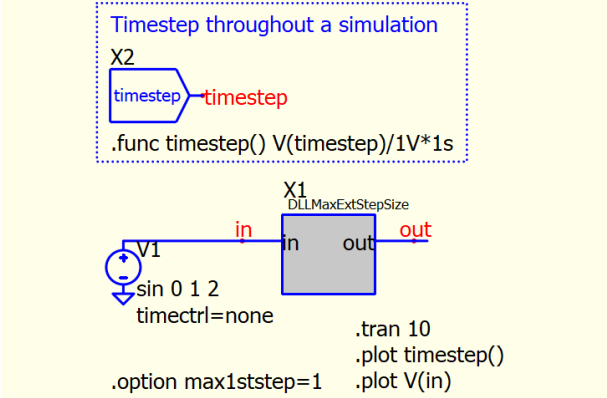



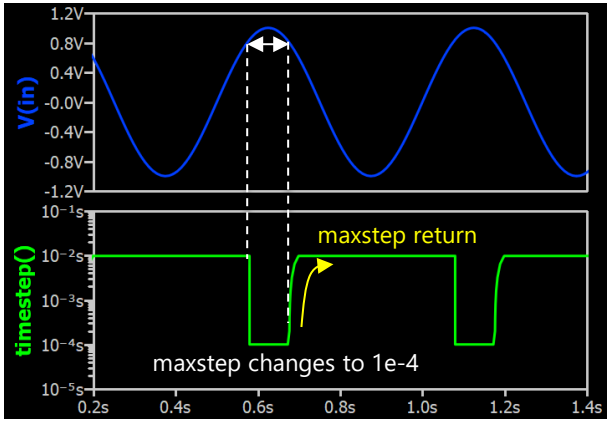



# How timestep works?

## Qspice : timestep - MaxExtStepSize.qsch

- #6 MaxExtStepSize (DLL)
  - MaxExtStepSize() is a function in DLL device
  - It allows a structure variable to be passed in order to control the maximum timestep
  - The return value of MaxExtStepSize() will determine the maxstep value
  - In this example
    - Target maximum step is determined by condition explained in #1b, which is 10s/1000=1e-2=$10^{-2}$s
    - In the DLL, MaxExtStepSize() reduces maxstep to 1e-4=$10^{-4}$s when V(in) > 0.8



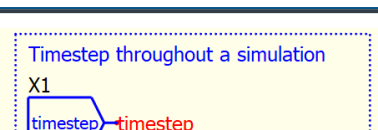

```
struct sDLLMAXEXTSTEPSIZE
{
  // declare the structure here
  float x;
};

extern "C" __declspec(dllexport) void dllmaxextstepsize(struct sDLLMAXEXTST
{
   double  in  = data[0].d; // input
   double &out = data[1].d; // output

   if(!*opaque)
   {
      *opaque = (struct sDLLMAXEXTSTEPSIZE *) malloc(sizeof(struct sDLLMAXE
      bzero(*opaque, sizeof(struct sDLLMAXEXTSTEPSIZE));
   }
   struct sDLLMAXEXTSTEPSIZE *inst = *opaque;

// Implement module evaluation code here:
   out = in;
   inst->x = in;
}

extern "C" __declspec(dllexport) double MaxExtStepSize(struct sDLLMAXEXTSTE
{
   if (inst->x >= 0.8)
      return 1e-4;
   return 1e308; // implement a good choice of max timestep size that deper
}
```

# How timestep works? (TTOL Devices) – Switch as example

## Qspice : timestep - SW TTOL.qsch

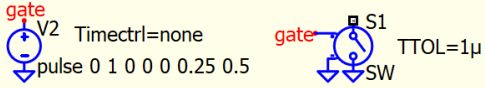
- TTOL Temporal Tolerance
  - TTOL is used in Switch, ¥-Device, Ø-Device etc…
  - In Ø-Device, user can control when to trigger *timestep=ttol in the Trunc() function
  - The Trunc() function in the TTOL device is implemented in a meaningful way to detect if the state has changed at the future simulation step (current simulation time + next timestep)
  - If the future state, when compared to the current state, is found to have changed in the TTOL device, the *timestep=TTOL is assigned, forcing the next step to only increase by the value of TTOL
  - Following simulation will increase each step by the active timestep multiplied by 2
    - If a state change is detected again, the timestep will be reset to TTOL once more
    - If no state change is detected, the timestep will continue to increase by the active timestep multiplied by 2 until it reaches the simulation step determined by Qspice based on the simulation setup

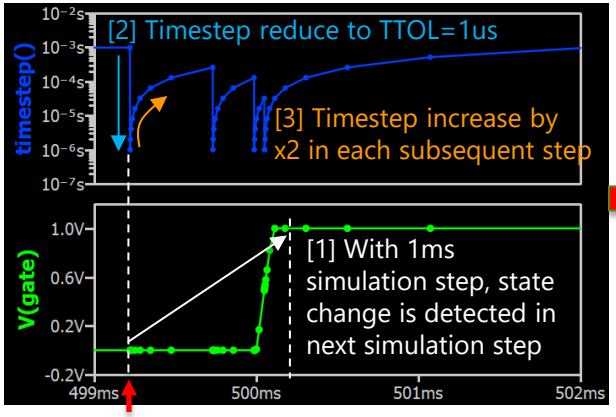
Timestep throughout a simulation

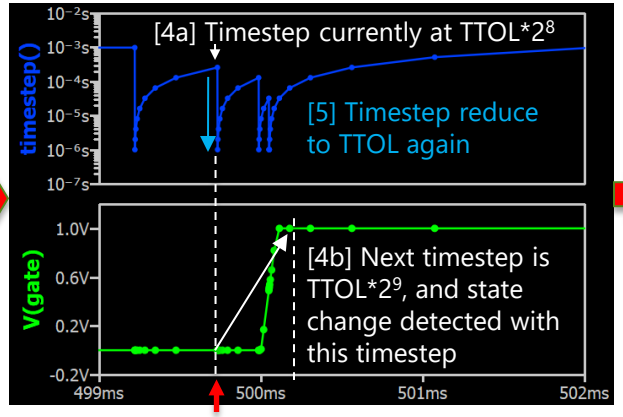


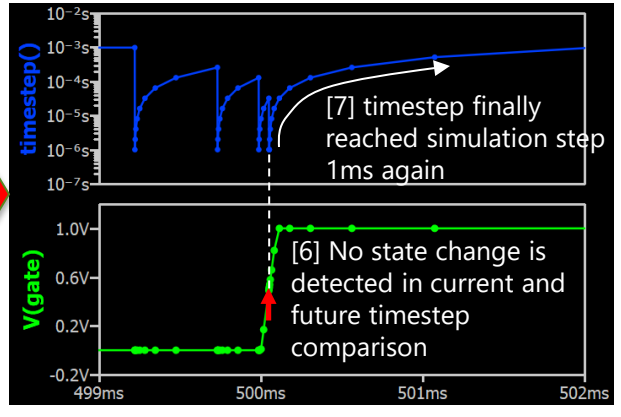

Simulation Circuit

```
.tran 0 Tstop Tstart
.plot V(gate)
.plot timestep()
.param Tstart 0  .param Tstop 1
.step dec param Tstop 1p 1K 10
.meas timestep max V(Tstep)
```



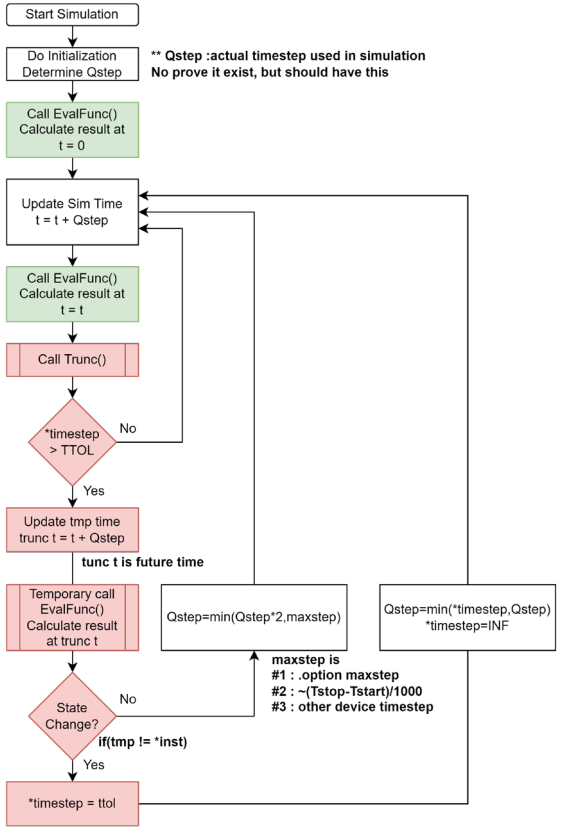



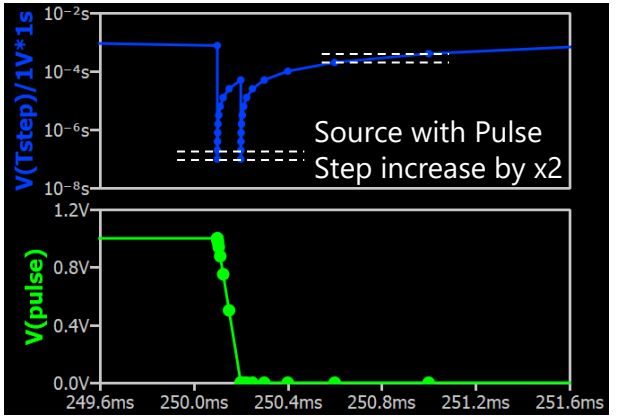



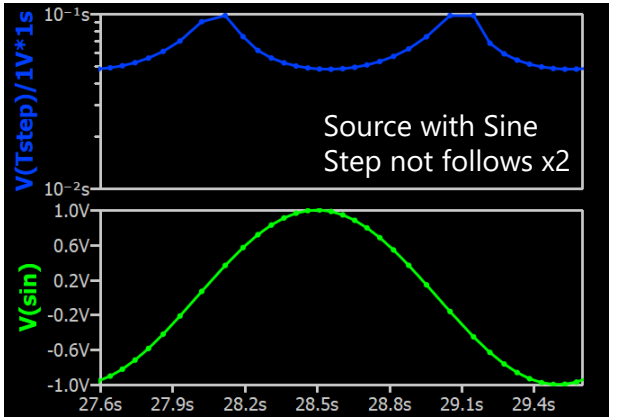
# How timestep works? (TTOL Devices) – Switch as example

## Qspice : timestep - Pulse Timectrl.qsch | timestep - Sin Timectrl.qsch

- TTOL Temporal Tolerance
  - Qspice employs different timestep control scheme. For example, source with sine doesn't follow x2 timestep relationship as TTOL does (shown in plot)
  - Qspice does not go back in time during simulation but instead looks at the future step to determine whether it needs to reduce the next simulation timestep
  - If the future step causes a state change, Qspice recognizes that the current timestep is not suitable and reduces its timestep to TTOL
  - Once TTOL is triggered, every subsequent timestep is multiplied by 2 until it reaches the maximum step condition



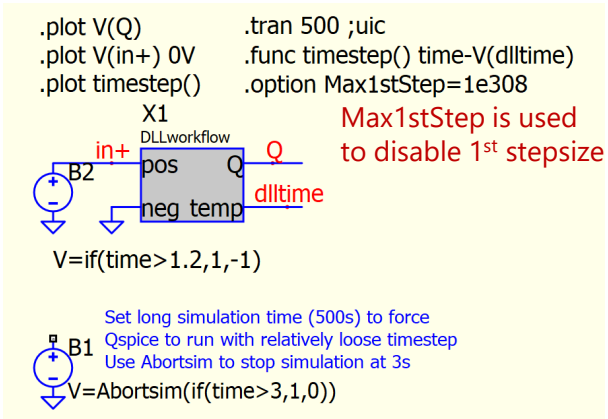



# How timestep works? (DLL Ø-Device)

## Qspice : DLLworkflow.qsch | dllworkflow.cpp

- DLL workflow (analysis code)
  - C++ code with multiple display to return t, *timestep at moment includes
    - main : standard main call
    - trunc-main : main called from Trunc()
    - trunc-enter : just enter Trunc()
    - trunc-1st if : just after if(*timestep>ttol) is TRUE
    - trunc-2nd if : just after if(tmp!=*inst) is TRUE
    - trunc-leave : before leaving Trunc()
  - Major variable
    - inst->tmain : dll time (t)
  - Special setup in schematic
    - Setup .tran to 500s but abortsim at 3s, to force Qspice to default maxstep ~ 500ms
    - Timestep is calculated with analog time – DLL time with .func timestep()




```
// Implement module evaluation code here:
   Q = pos > neg;
   temp = t;
   inst->lastQ = Q;
   inst->tmain = t;
   if (inst->inTrunc == 0)
      display("main             : t=%.12f\r\n",t);
   else
      display("  trunc - main   : t=%.12f\r\n",t);
}
```

```
extern "C" __declspec(dllexport) void Trunc(struc
{ // limit the timestep to a tolerance if the cir
   const double ttol = 1e-3;
   //const double ttol = 1;
   display("  trunc - enter : t=%.12f  *timestep=
   if(*timestep > ttol)
   {
      display("  trunc - 1st IF: t=%.12f  *timest
      bool   &Q    = data[2].b; // output
      double &temp = data[3].d; // output

      // Save output vector
      const bool   _Q    = Q   ;
      const double _temp = temp;

      inst->inTrunc=1;
      struct sDLLWORKFLOW tmp = *inst;
      dllworkflow(&(&tmp), t, data);
      inst->inTrunc=0;
   // if(tmp != *inst) // implement a meaningful
   //    *timestep = ttol;
      if(tmp.lastQ != inst->lastQ){
         *timestep = ttol;
         display("  trunc - 2nd IF: t=%.12f  *tim
      }

      // Restore output vector
      Q    = _Q   ;
      temp = _temp;
   }
   display("  trunc - leave : t=%.12f  *timestep=
}
```

Inst->inTrunc = 1 if main is called from Trunc()

# How timestep works? (DLL Ø-Device) : TTOL=1e-3 in Trunc()

## Qspice : DLLworkflow.qsch | dllworkflow.cpp

```
C:\KSKelvinQspice\02 Advance Topics\How Time Step Works in Qspice\dll_workflow\DLLworkflow.qsch
main            : t=0.000000000000
main            : t=0.000000000000
main            : t=0.000000000000
main            : t=0.000000000000
main            : t=0.000000000000
  trunc - enter : t=0.000000000000  *timestep=inf  inst->tmain=0.000000000000  dT=0.000000000000
  trunc - 1st IF: t=0.000000000000  *timestep=inf
  trunc - main  : t=0.000000000000
  trunc - leave : t=0.000000000000  *timestep=inf
main            : t=0.000000000000
  trunc - enter : t=0.488281250000  *timestep=inf  inst->tmain=0.000000000000  dT=0.488281250000
  trunc - 1st IF: t=0.488281250000  *timestep=inf
  trunc - main  : t=0.488281250000
  trunc - leave : t=0.488281250000  *timestep=inf
main            : t=0.488281250000
  trunc - enter : t=0.976562500000  *timestep=inf  inst->tmain=0.488281250000  dT=0.488281250000
  trunc - 1st IF: t=0.976562500000  *timestep=inf
  trunc - main  : t=0.976562500000
  trunc - leave : t=0.976562500000  *timestep=inf
  trunc - enter : t=0.976562500000  *timestep=inf  inst->tmain=0.488281250000  dT=0.488281250000
  trunc - 1st IF: t=0.976562500000  *timestep=inf
  trunc - main  : t=0.976562500000
  trunc - leave : t=0.976562500000  *timestep=inf
main            : t=0.976562500000
  trunc - enter : t=1.464843750000  *timestep=inf  inst->tmain=0.976562500000  dT=0.488281250000
  trunc - 1st IF: t=1.464843750000  *timestep=inf
  trunc - main  : t=1.464843750000
  trunc - 2nd IF: t=1.464843750000  *timestep=0.001000000000
  trunc - leave : t=1.464843750000  *timestep=0.001000000000
  trunc - enter : t=0.977562500000  *timestep=inf  inst->tmain=0.976562500000  dT=0.001000000000
  trunc - 1st IF: t=0.977562500000  *timestep=inf
  trunc - main  : t=0.977562500000
  trunc - leave : t=0.977562500000  *timestep=inf
  trunc - enter : t=0.977562500000  *timestep=inf  inst->tmain=0.976562500000  dT=0.001000000000
  trunc - 1st IF: t=0.977562500000  *timestep=inf
  trunc - main  : t=0.977562500000
  trunc - leave : t=0.977562500000  *timestep=inf
main            : t=0.977562500000
  trunc - enter : t=0.979562500000  *timestep=inf  inst->tmain=0.977562500000  dT=0.002000000000
  trunc - 1st IF: t=0.979562500000  *timestep=inf
  trunc - main  : t=0.979562500000
  trunc - leave : t=0.979562500000  *timestep=inf
  trunc - enter : t=0.979562500000  *timestep=inf  inst->tmain=0.977562500000  dT=0.002000000000
  trunc - 1st IF: t=0.979562500000  *timestep=inf
  trunc - main  : t=0.979562500000
  trunc - leave : t=0.979562500000  *timestep=inf
main            : t=0.979562500000
  trunc - enter : t=0.983562500000  *timestep=inf  inst->tmain=0.979562500000  dT=0.004000000000
  trunc - 1st IF: t=0.983562500000  *timestep=inf
  trunc - main  : t=0.983562500000
  trunc - leave : t=0.983562500000  *timestep=inf
main            : t=0.983562500000
  trunc - enter : t=0.991562500000  *timestep=inf  inst->tmain=0.983562500000  dT=0.008000000000
  trunc - 1st IF: t=0.991562500000  *timestep=inf
  trunc - main  : t=0.991562500000
  trunc - leave : t=0.991562500000  *timestep=inf
main            : t=0.991562500000
  trunc - enter : t=1.007562500000  *timestep=inf  inst->tmain=0.991562500000  dT=0.016000000000
```

@t=0, trunc() tested next step and no state change is detected

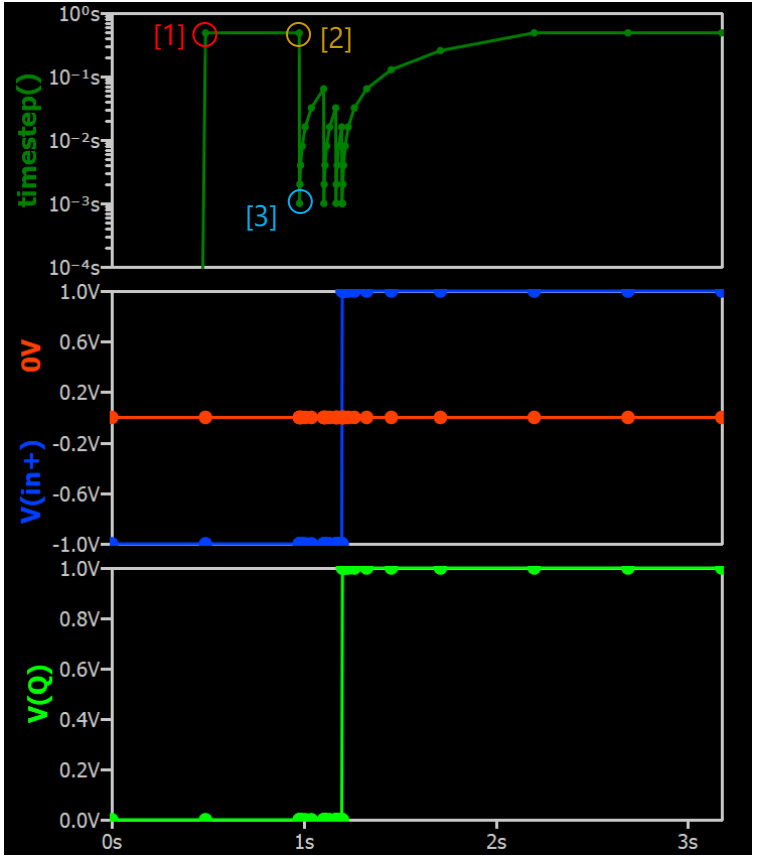
[1] Trunc() executed two times between each main() if no *temistep is assigned

[3] if(*tmp!=inst) is FALSE, no *timestep is assigned → next sim time in main() is this future test time

[2] if(*tmp!=inst) is TRUE, *timestep=TTOL t=1.4648... is future test time in Trunc()

[3] t=1.4648... is ignored after *timestep set to TTOL → forces next sim time t=0.9765... +ttol=0.9775...

[4] Timestep multiplier by two in subsequent TTOL event



# How timestep works? (DLL Ø-Device) : TTOL=1 in Trunc()

## Qspice : DLLworkflow.qsch | dllworkflow.cpp

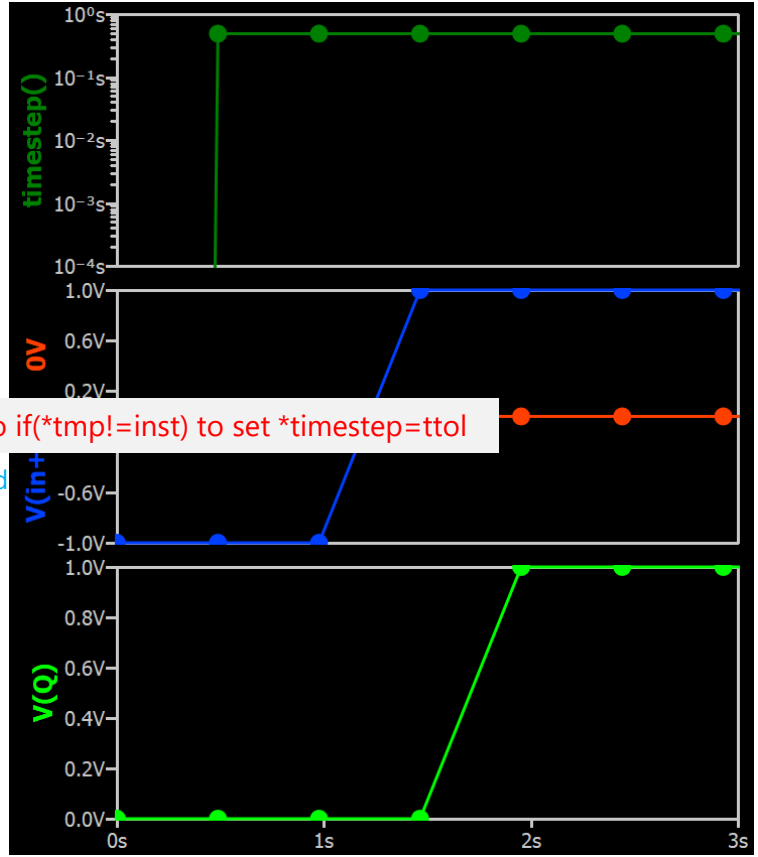
** TTOL is larger than timestep using in this simulation

```
C:\KSKelvinQspice\02 Advance Topics\How Time Step Works in Qspice\dll_workflow\DLLworkflow.qsch
main            : t=0.000000000000
main            : t=0.000000000000
main            : t=0.000000000000
main            : t=0.000000000000
main            : t=0.000000000000
  trunc - enter : t=0.000000000000  *timestep=inf  inst->tmain=0.000000000000  dT=0.000000000000
  trunc - 1st IF: t=0.000000000000  *timestep=inf
  trunc - main  : t=0.000000000000
  trunc - leave : t=0.000000000000  *timestep=inf
main            : t=0.000000000000
  trunc - enter : t=0.488281250000  *timestep=inf  inst->tmain=0.000000000000  dT=0.488281250000
  trunc - 1st IF: t=0.488281250000  *timestep=inf
  trunc - main  : t=0.488281250000
  trunc - leave : t=0.488281250000  *timestep=inf
main            : t=0.488281250000
  trunc - enter : t=0.976562500000  *timestep=inf  inst->tmain=0.488281250000  dT=0.488281250000
  trunc - 1st IF: t=0.976562500000  *timestep=inf
  trunc - main  : t=0.976562500000
  trunc - leave : t=0.976562500000  *timestep=inf
  trunc - enter : t=0.976562500000  *timestep=inf  inst->tmain=0.488281250000  dT=0.488281250000
  trunc - 1st IF: t=0.976562500000  *timestep=inf
  trunc - main  : t=0.976562500000
  trunc - leave : t=0.976562500000  *timestep=inf
main            : t=0.976562500000
  trunc - enter : t=1.464843750000  *timestep=inf  inst->tmain=0.976562500000  dT=0.488281250000
  trunc - 1st IF: t=1.464843750000  *timestep=inf
  trunc - main  : t=1.464843750000
  trunc - 2nd IF: t=1.464843750000  *timestep=1.000000000000
  trunc - leave : t=1.464843750000  *timestep=1.000000000000
  trunc - enter : t=1.464843750000  *timestep=inf  inst->tmain=0.
  trunc - 1st IF: t=1.464843750000  *timestep=inf
  trunc - main  : t=1.464843750000
  trunc - 2nd IF: t=1.464843750000  *timestep=1.000000000000
  trunc - leave : t=1.464843750000  *timestep=1.000000000000
main            : t=1.464843750000
  trunc - enter : t=1.953125000000  *timestep=inf  inst->tmain=1.464843750000  dT=0.488281250000
  trunc - 1st IF: t=1.953125000000  *timestep=inf
  trunc - main  : t=1.953125000000
  trunc - leave : t=1.953125000000  *timestep=inf
  trunc - enter : t=1.953125000000  *timestep=inf  inst->tmain=1.464843750000  dT=0.488281250000
  trunc - 1st IF: t=1.953125000000  *timestep=inf
  trunc - main  : t=1.953125000000
  trunc - leave : t=1.953125000000  *timestep=inf
main            : t=1.953125000000
  trunc - enter : t=2.441406250000  *timestep=inf  inst->tmain=1.953125000000  dT=0.488281250000
  trunc - 1st IF: t=2.441406250000  *timestep=inf
  trunc - main  : t=2.441406250000
  trunc - leave : t=2.441406250000  *timestep=inf
main            : t=2.441406250000
  trunc - enter : t=2.929687500000  *timestep=inf  inst->tmain=2.441406250000  dT=0.488281250000
  trunc - 1st IF: t=2.929687500000  *timestep=inf
  trunc - main  : t=2.929687500000
  trunc - leave : t=2.929687500000  *timestep=inf
main            : t=2.929687500000
  trunc - enter : t=3.417968750000  *timestep=inf  inst->tmain=2.929687500000  dT=0.488281250000
```

[1] Same as previous up to this line

[2] As *timestep is INF, still goto if(*tmp!=inst) to set *timestep=ttol

[3] t=1.464... is still effective and ignored *timestep=ttol=1 assignment



# Conclusion

- Conclusion from this Study
  - Timestep in Qspice is adaptive, which determine by
    - .option maxstep
    - .option max1ststep : the first timestep in .tran
    - Tstart and Tstop in .tran
    - Devices with timestep control ability (e.g. Voltage source, Switch, ¥-Device)
    - Return of MaxExtStepSize() or *timestep in Trunc() in DLL block (Ø-Device)
  - Qspice never goes back in time during simulation, but it examines the future steps to determine if the timestep is too aggressive based on user-defined criteria in DLL block
  - Qspice devices can utilize output state changes with if(*tmp!=inst) OR whatever you do to force *timestep to change within Trunc().
  - *timestep in Trunc() is always equal +INF when just enter Trunc(). It seems if condition (*timestep>TTOL) is always TRUE in Trunc()
    - *timestep in Trunc() not actual timestep itself but a determination factor for actual timestep
  - If trunc() exit with *timestep change, next simulation time will force to increase by this amount of change (but with exception that *timestep > ~(Tstop-Tstart)/1000)
  - The actual timestep will be increased by a factor of 2 in each subsequent step, until
    - Re-trigger of if(*tmp!=inst) OR
    - Reach ~(Tstop-Tstart)/1000 OR
    - Timestep limit from other devices

# Appendix : How timestep works? (DLL Ø-Device) – First Step in DLL

## Qspice : DLLworkflow - 1st Trunc.qsch

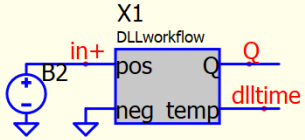
```
C:\KSKelvinQspice\02 Advance Topics\How Time Step Works in Qspice\trunc_1st\DLLworkflow - 1st Trur
main            : t=0.000000000000
main            : t=0.000000000000
main            : t=0.000000000000
main            : t=0.000000000000
main            : t=0.000000000000
  trunc - enter : t=0.000000000000  *timestep=inf  inst->tmain=0.000000000000  dT=0.000000000000
  trunc - 1st IF: t=0.000000000000  *timestep=inf
  trunc - main  : t=0.000000000000
  trunc - leave : t=0.000000000000  *timestep=inf
main            : t=0.000000000000
  trunc - enter : t=0.000000097656  *timestep=inf  inst->tmain=0.000000000000  dT=0.000000097656
  trunc - 1st IF: t=0.000000097656  *timestep=inf
  trunc - main  : t=0.000000097656
  trunc - 2nd IF: t=0.000000097656  *timestep=0.000000001000
  trunc - leave : t=0.000000097656  *timestep=0.000000001000
  trunc - enter : t=0.000000048828  *timestep=inf  inst->tmain=0.000000000000  dT=0.000000048828
  trunc - 1st IF: t=0.000000048828  *timestep=inf
  trunc - main  : t=0.000000048828
  trunc - 2nd IF: t=0.000000048828  *timestep=0.000000001000
  trunc - leave : t=0.000000048828  *timestep=0.000000001000
  trunc - enter : t=0.000000024414  *timestep=inf  inst->tmain=0.000000000000  dT=0.000000024414
  trunc - 1st IF: t=0.000000024414  *timestep=inf
  trunc - main  : t=0.000000024414
  trunc - leave : t=0.000000024414  *timestep=inf
main            : t=0.000000024414
  trunc - enter : t=0.000000048828  *timestep=inf  inst->tmain=0.000000024414  dT=0.000000024414
  trunc - 1st IF: t=0.000000048828  *timestep=inf
  trunc - main  : t=0.000000048828
  trunc - 2nd IF: t=0.000000048828  *timestep=0.000000001000
  trunc - leave : t=0.000000048828  *timestep=0.000000001000
  trunc - enter : t=0.000000025414  *timestep=inf  inst->tmain=0.000000024414  dT=0.000000001000
  trunc - 1st IF: t=0.000000025414  *timestep=inf
  trunc - main  : t=0.000000025414
  trunc - 2nd IF: t=0.000000025414  *timestep=0.000000001000
  trunc - leave : t=0.000000025414  *timestep=0.000000001000
  trunc - enter : t=0.000000025414  *timestep=inf  inst->tmain=0.000000024414  dT=0.000000001000
  trunc - 1st IF: t=0.000000025414  *timestep=inf
  trunc - main  : t=0.000000025414
  trunc - 2nd IF: t=0.000000025414  *timestep=0.000000001000
  trunc - leave : t=0.000000025414  *timestep=0.000000001000
main            : t=0.000000025414
  trunc - enter : t=0.000000026414  *timestep=inf  inst->tmain=0.000000025414  dT=0.000000001000
```

These only exist if uic is in .tran

@t=0 in main, trunc() test future step at t=9.7656e-8

Use binary search for the first timestep size if *timestep is assigned

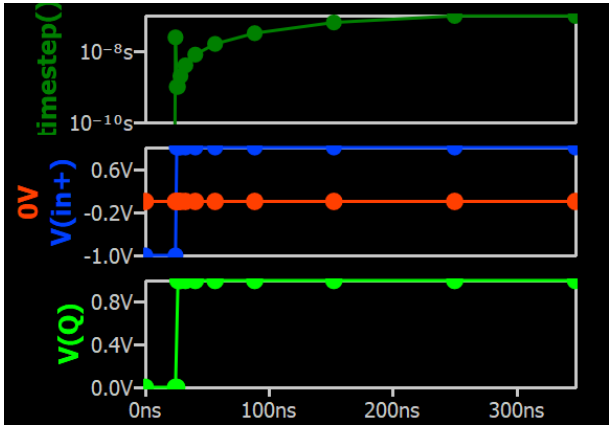
```
.plot V(Q)                .tran 100n*1000 ;uic
.plot V(in+) 0V           .func timestep() time-V(dlltime)
.plot timestep()          .option Max1stStep=10n
```




V=if(time>25n,1,-1)


Set long simulation time (500s) to force Qspice to run with relatively loose timestep
Use Abortsim to stop simulation at 3s

V=Abortsim(if(time>300n,1,0))



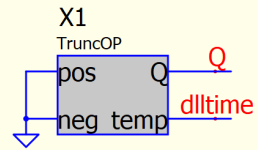
# Appendix A

# *timestep in TRUNC

# *timestep=TTOL in Trunc()

## Qspice : TruncOP.qsch | truncop.dll

```
.tran 500 ;uic
.func timestep() time-V(dlltime)
.plot timestep()
                    ; disable first-step limit
                    .option MAX1STSTEP=1e308
```




Set long simulation time (500s) to force Qspice to run with relatively loose timestep
Use Abortsim to stop simulation at 3s

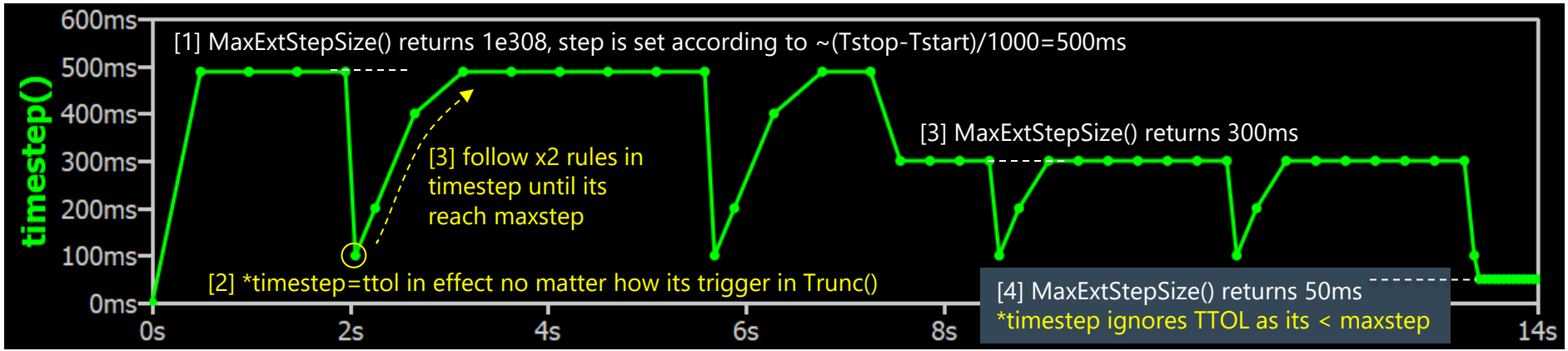

```
// Implement module evaluation code here:
   temp = t;
   inst->tmain = t;
   inst->counter++;
   if (inst->inTrunc == 0)
   {
      display("main             : t=%.12f\r\n",t);
   }
   else
      display("  trunc - main   : t=%.12f\r\n",t);
}

extern "C" __declspec(dllexport) double MaxExtStepSize
{
   if(inst->counter > 50)
      return 5e-2;
   else if (inst->counter > 25)
      return 3e-1;
   else
      return 1e308; // implement a good choice of max
}
```

Change maxstep at different count

**Test concept**

- A counter in module evaluation code
- Whatever counter%10==0, trigger *timestep=ttol in Trunc()

```
extern "C" __declspec(dllexport) void Trunc(
{ // limit the timestep to a tolerance if th
   const double ttol = 1e-1;

// if(tmp != *inst) // implement
//    *timestep = ttol;
   if(inst->counter%10==0)
   {
      *timestep = ttol;
      display("  trunc - 2nd IF:
      inst->counter++;
   }
```

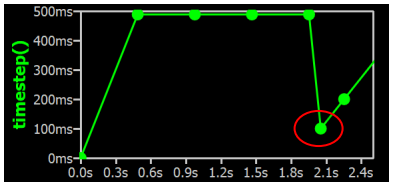


# Special Obervation (With and Without counter++ in Trunc())

## [No Explanation yet]

### With inst->counter++

```
if(inst->counter%10==0)
{
    *timestep = ttol;
    display("  trunc - 2n
    inst->counter++;  //
}
```



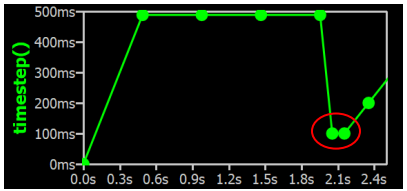

```
  trunc - leave : t=1.953125000000  *timestep=inf
main            : t=1.953125000000
  trunc - enter : t=2.441406250000  *timestep=inf  inst->tmain=1.953125000000  dT=0.488281250000
  trunc - 1st IF: t=2.441406250000  *timestep=inf
  trunc - main  : t=2.441406250000
  trunc - 2nd IF: t=2.441406250000  *timestep=0.100000000000
  trunc - leave : t=2.441406250000  *timestep=0.100000000000
  trunc - enter : t=2.053125000000  *timestep=inf  inst->tmain=1.953125000000  dT=0.100000000000
  trunc - 1st IF: t=2.053125000000  *timestep=inf
  trunc - main  : t=2.053125000000
  trunc - leave : t=2.053125000000  *timestep=inf
main            : t=2.053125000000
  trunc - enter : t=2.253125000000  *timestep=inf  inst->tmain=2.053125000000  dT=0.200000000000
  trunc - 1st IF: t=2.253125000000  *timestep=inf
  trunc - main  : t=2.253125000000
  trunc - leave : t=2.253125000000  *timestep=inf
main            : t=2.253125000000
  trunc - enter : t=2.653125000000  *timestep=inf  inst->tmain=2.253125000000  dT=0.400000000000
  trunc - 1st IF: t=2.653125000000  *timestep=inf
  trunc - main  : t=2.653125000000
  trunc - leave : t=2.653125000000  *timestep=inf
main            : t=2.653125000000
  trunc - enter : t=3.141406250000  *timestep=inf  inst->tmain=2.653125000000  dT=0.488281250000
  trunc - 1st IF: t=3.141406250000  *timestep=inf
  trunc - main  : t=3.141406250000
  trunc - leave : t=3.141406250000  *timestep=inf
```

Identical before this line

1st step is same by t+ttol

### Without inst->counter++

```
if(inst->counter%10==0)
{
    *timestep = ttol;
    display("  trunc - 2
    //inst->counter++;
}
```




```
  trunc - leave : t=1.953125000000  *timestep=inf
main            : t=1.953125000000
  trunc - enter : t=2.441406250000  *timestep=inf  inst->tmain=1.953125000000  dT=0.488281250000
  trunc - 1st IF: t=2.441406250000  *timestep=inf
  trunc - main  : t=2.441406250000
  trunc - 2nd IF: t=2.441406250000  *timestep=0.100000000000
  trunc - leave : t=2.441406250000  *timestep=0.100000000000
  trunc - enter : t=2.053125000000  *timestep=inf  inst->tmain=1.953125000000  dT=0.100000000000
  trunc - 1st IF: t=2.053125000000  *timestep=inf
  trunc - main  : t=2.053125000000
  trunc - 2nd IF: t=2.053125000000  *timestep=0.100000000000
  trunc - leave : t=2.053125000000  *timestep=0.100000000000
main            : t=2.053125000000
  trunc - enter : t=2.153125000000  *timestep=inf  inst->tmain=2.053125000000  dT=0.100000000000
  trunc - 1st IF: t=2.153125000000  *timestep=inf
  trunc - main  : t=2.153125000000
  trunc - leave : t=2.153125000000  *timestep=inf
main            : t=2.153125000000
  trunc - enter : t=2.353125000000  *timestep=inf  inst->tmain=2.153125000000  dT=0.200000000000
  trunc - 1st IF: t=2.353125000000  *timestep=inf
  trunc - main  : t=2.353125000000
  trunc - leave : t=2.353125000000  *timestep=inf
main            : t=2.353125000000
  trunc - enter : t=2.753125000000  *timestep=inf  inst->tmain=2.353125000000  dT=0.400000000000
  trunc - 1st IF: t=2.753125000000  *timestep=inf
  trunc - main  : t=2.753125000000
  trunc - leave : t=2.753125000000  *timestep=inf
```

If(inst->counter%10==0) is still true in 2nd trunc() test
*timestep=ttol is re-assign again

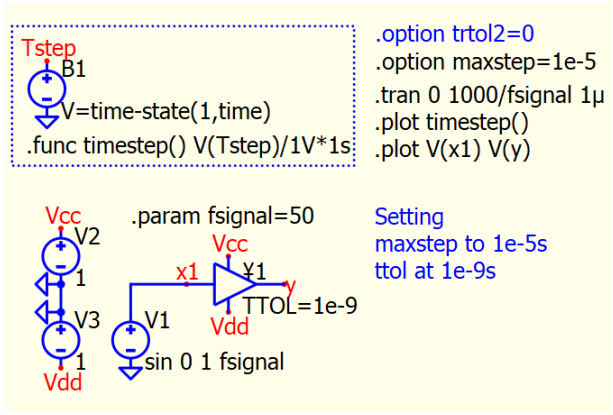
# Appendix B

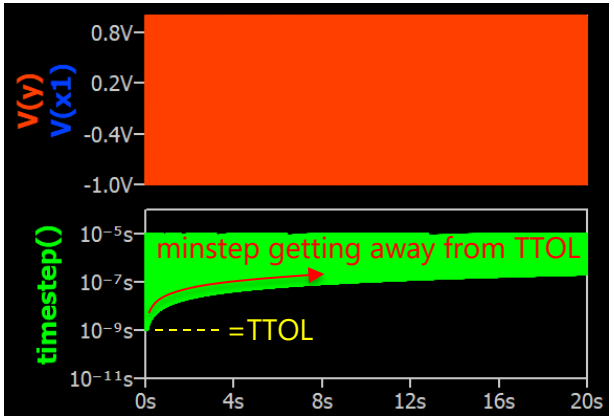
# Simulation with Long Run

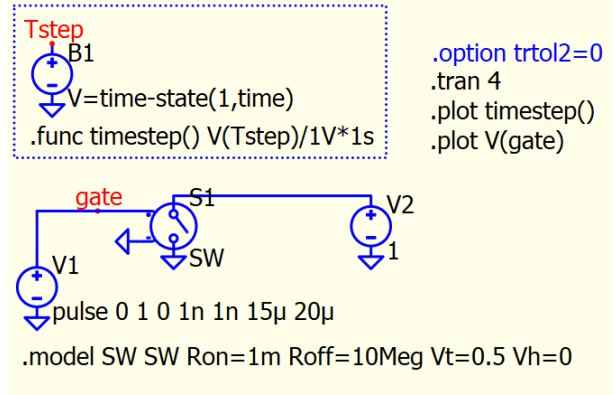
# timestep÷time limitation

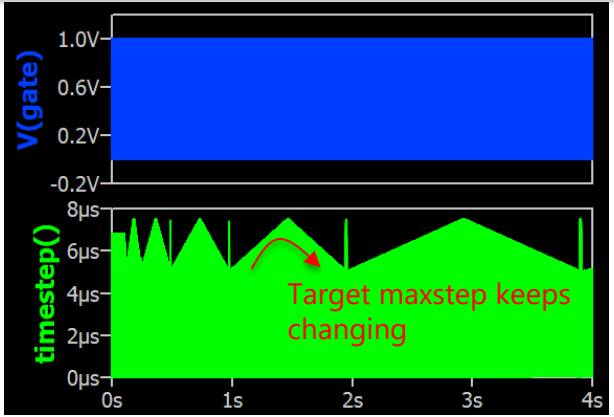
## Qspice : timestep-ttol.qsch | timestep-maxstep.qsch

- timestep÷time limitation
  - By Mike Engerhardt, Qspice cannot go to tiny time steps are latetimes because it might have not enough resolution (i.e. timestep÷time has to be several orders of magnitude larger than 1e-15 so that it can do the math)
  - First example, a device with TTOL is used in a long simulation run, and the minimum timestep gradually increases over time
  - Second example, a V-source and switch with default timectrl TTOL in a long simulation run, and the maxstep keeps changing throughout the simulation



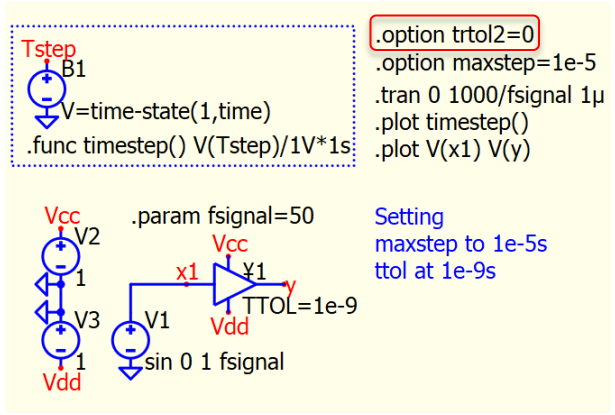



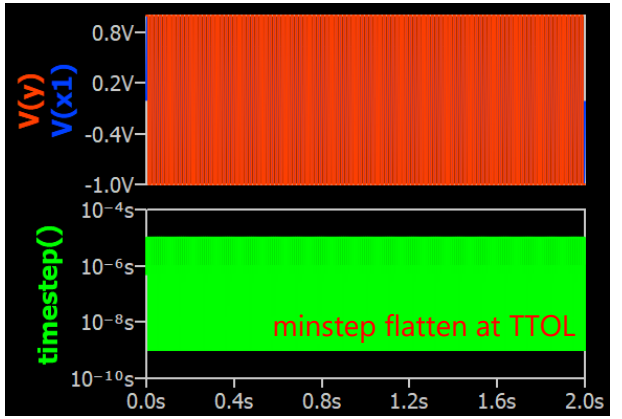



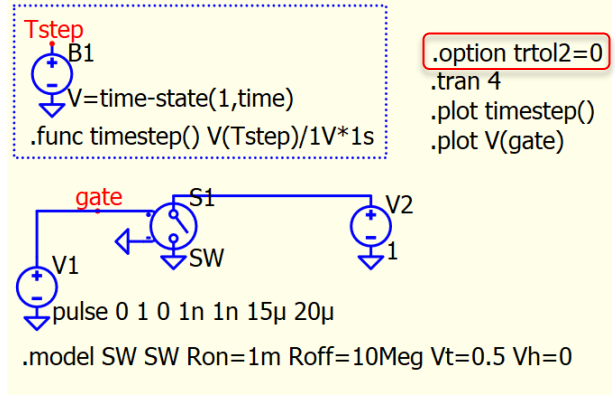
# TRTOL2 in timestep÷time limited situation

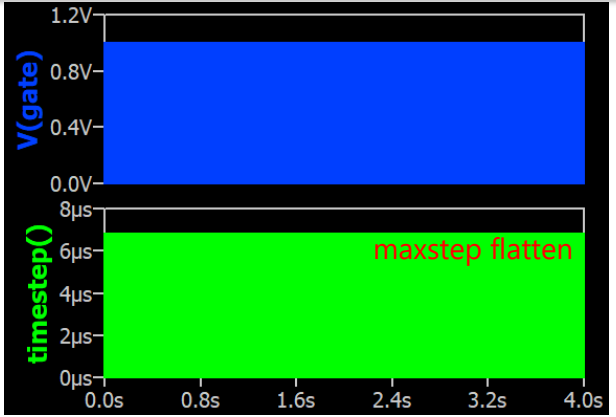
## Qspice : timestep-ttol.qsch | timestep-maxstep.qsch

- TRTOL2
  - Trtol2 : Another dimensionless truncation error guidance
  - **Default TRTOL2=1e-8**
  - It can observe that by focusing TRTOL2=0 can flatten maxstep and minstep along simulation
  - Quote from Mike Engerhardt, TRTOL2 is Qspice option to prevents the simulation from crashing by going to a smaller timestep that is actually required



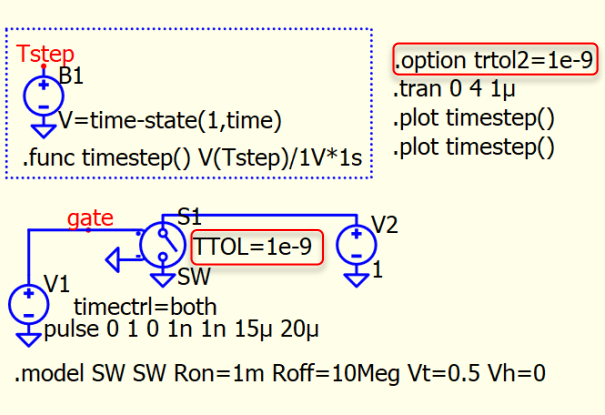



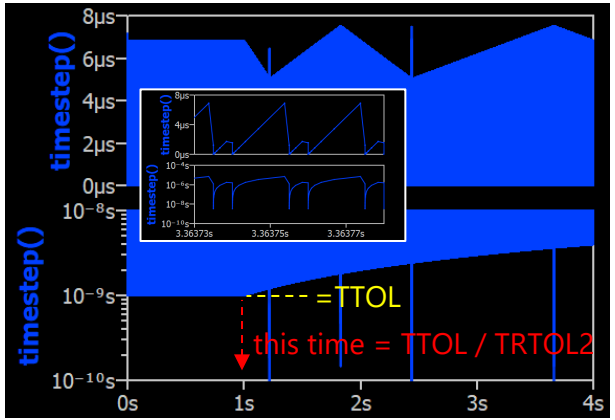



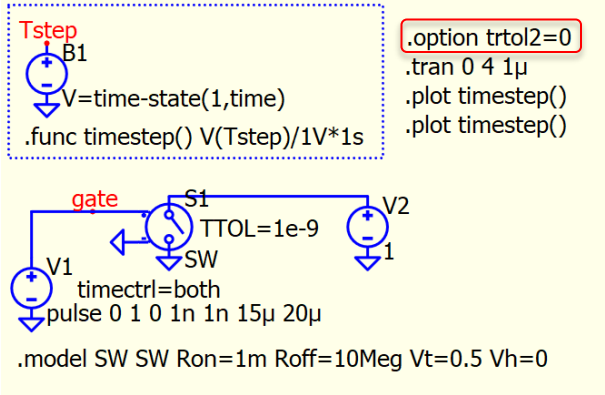
# Study of TRTOL2

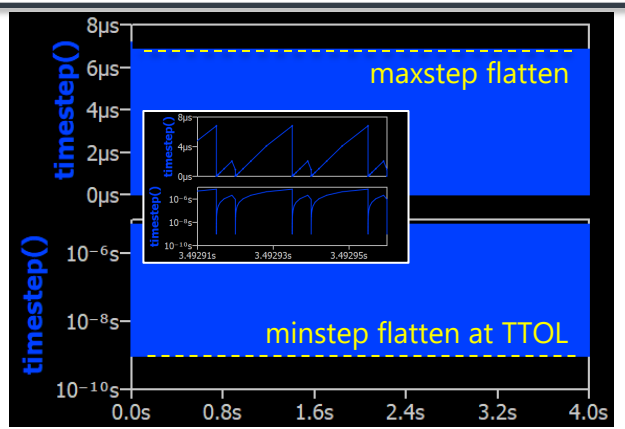
## Qspice : timestep-trtol2.qsch

- Study of TRTOL2
  - This simulation setup with V-source timectrl and TTOL both works together
  - It is observed that the timestep starts failing from TTOL from **simulation time = TTOL / TRTOL2**
  - Therefore, forcing TRTOL2=0 will extend this time to infinite and maxstep and minstep both flatten across entire simulation



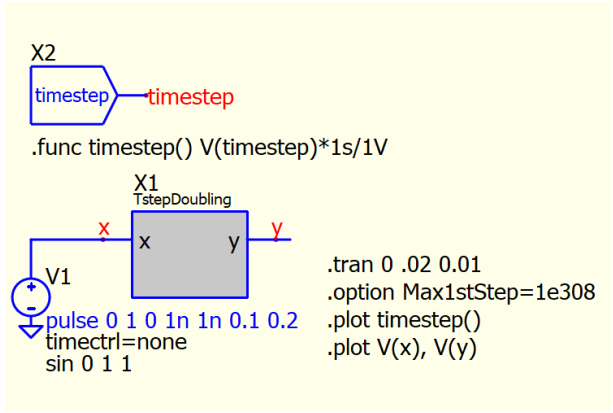
# Appendix C

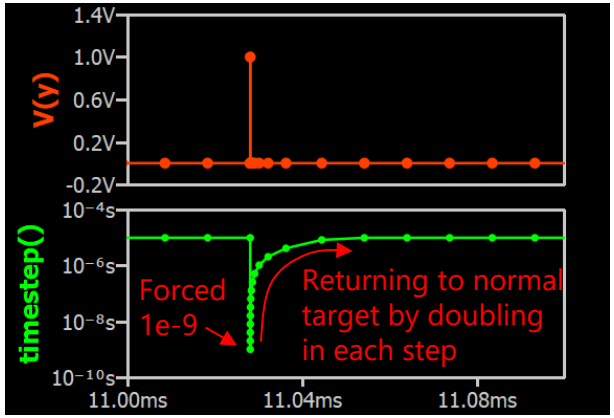
# Timestep Doubling in Qspice

# Timestep Doubling in MaxExtStepSize() function

## Qspice : TstepDoubling.qsch | tstepdoubling.cpp

- Timestep Doubling
  - Whether you use Trunc() or MaxExtStepSize() to set the timestep, it will trigger the timestep doubling algorithm to adjust the timestep back to the desired step size
  - This example forces MaxExtStepSize() to set a timestep of 1e-9 every 0.001s. It confirms that when the timestep is no longer forced to be 1e-9, it will return to the desired step size using a timestep doubling strategy




```
struct sTSTEPDOUBLING
{
  // declare the structure here
  float MaxStepTtol;
  float lastT;
  bool MaxStepTrig;
};

extern "C" __declspec(dllexport) void tstepdoubling(str
{
   double  x = data[0].d; // input
   double &y = data[1].d; // output

   if(!*opaque)
   {
      *opaque = (struct sTSTEPDOUBLING *) malloc(sizeof
      bzero(*opaque, sizeof(struct sTSTEPDOUBLING));
   }
   struct sTSTEPDOUBLING *inst = *opaque;

// Implement module evaluation code here:
   y = inst->MaxStepTrig;

   inst->MaxStepTtol = 1e-9;
   inst->MaxStepTrig = 0;
   if ( t - inst->lastT > 0.001 )
   {
      inst->MaxStepTrig = 1;
      inst->lastT = t;
   }
}

extern "C" __declspec(dllexport) double MaxExtStepSize(
{
   if (inst->MaxStepTrig)
      return inst->MaxStepTtol;
   else
      return 1e308; // implement a good choice of max t
}
```

About every 0.001s, set MaxStepTrig flag

Return timestep as MaxStepTtol (1e-9) from MaxExtStepSize() function

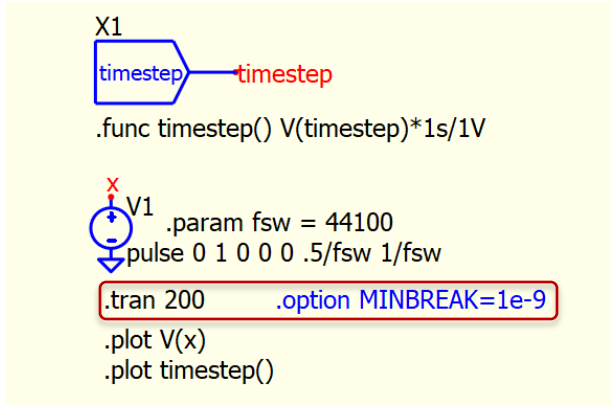
# Appendix D

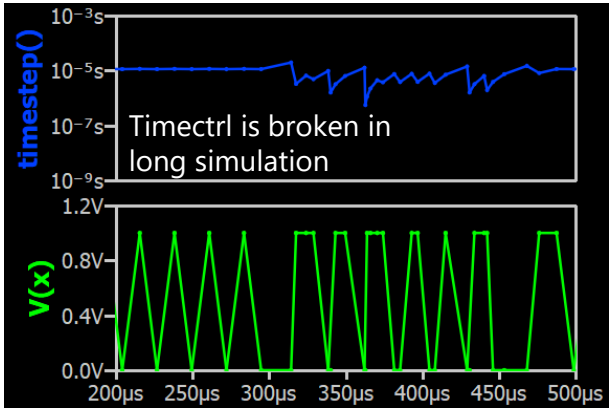
# MinBreak in Timectrl

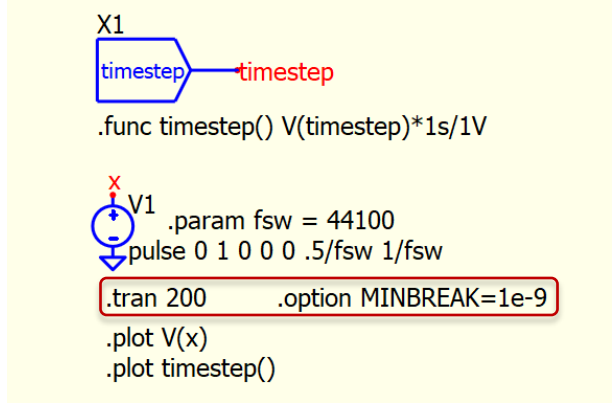
# MinBreak in Timectrl of V-source

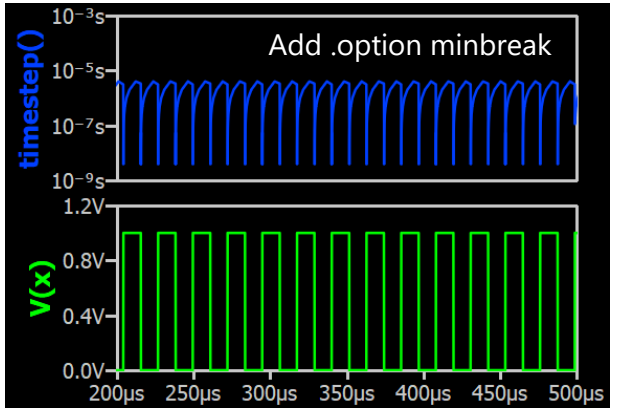
## Qspice : Option - Minbreak (.tran 200).qsch

- MinBreak in Timectrl
  - In long simulation run, timestep control of V-source may be broken in long simulation run
  - In this situation, several approaches can be considered
    - Add a 1pF capacitor in parallel to V-source, to limit timestep in slew
    - Add maxstep to limit maximum step
    - Add TTOL-device for TTOL timestep scheme
    - Add .option minbreak for minimum timestep in breakpoints for V-source



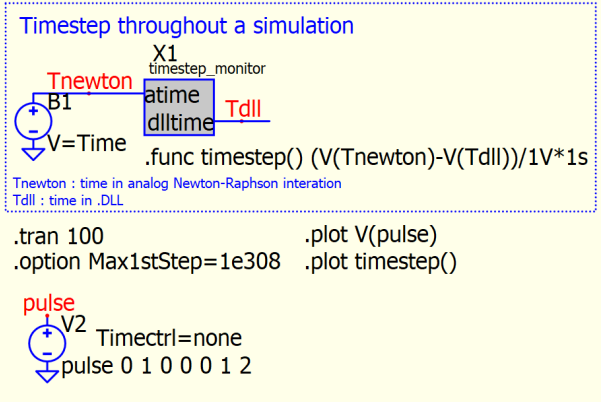
# Appendix E

# Timestep with DLL

# Appendix : How Time Step Works in Qspice – TimeStep Monitor (DLL)

## Qspice : timestep_monitor.qsch | timestep_monitor.cpp

- Timestep Monitor
  - Simulation Time of Qspice can be found as
    - Time in analog Newton-Raphson interation: Time
    - DLL Time : t in DLL block
  - DLL Time always one step behind Analog Time
    - Therefore, different of analog time and DLL time is simulation timestep
  - Method to read timestep
    - Cpp block with dlltime=atime, where atime is analog time and dlltime is dll time delayed by one timestep
    - Calculate different between analog and DLL time for timestep




- Code

```
// Implement module evaluation code here:
dlltime = atime;
```

  - only 1 line of code to pass time from input to output
  - as dll time (output) is always one step delay of input, different between dlltime and atime is timestep