

## AMBER Recipe Commands

---

**title** (*title*)

Displays the title of the growth in the title window during growth. This should be at or near the beginning of the recipe.

Example:

**title** (Doped superlattice structure)

Displays the title, "Doped superlattice structure" in the title window.

---

**comment** (*message*)

Displays a comment in the comment window during growth. This can be used to describe the current state of the growth. This is not the same as a ! command which is a way of marking a comment in the recipe.

Example:

**comment** (Buffer layer growth in progress)

Displays the comment "Buffer layer growth in progress" in the comment window

Example2:

**comment** ()

Clears the comment window.

---

!

Causes the recipe compiler to ignore the entire line.

Example:

```
! *****  
! *  
! * All of this text *  
! * is ignored by *  
! * the compiler. *  
! *  
! *****
```

---

```
wait [time in seconds]  
or  
wait [minutes:seconds]  
or  
wait [hours:minutes:seconds]
```

Pauses growth. After the specified time has elapsed, the growth continues immediately with the next instruction. While paused, the operator has the option of continuing before the time has elapsed, pausing beyond the originally specified time, or cancelling the remainder of the growth.

Example:

```
wait 4.5
```

Pauses growth for 4.5 seconds

Example:

```
wait 1:15
```

Pauses growth for 1 minute, 15 seconds

Example:

```
wait 2:05:10
```

Pauses growth for 2 hours, 5 minutes and 10 seconds

---

```
waituntil [time of day]  
or  
waituntil [time of day, day of the week]
```

Pauses until the specified time of day is reached. AM or PM must be included. Spaces between the time and am or pm are allowed. AM and PM are case insensitive. Optionally, the day may be included after a comma. The abbreviations for the days are: sun, mon, tue, wed, thu, fri, sat.

Example:

```
waituntil 6:00 AM
```

Pauses growth until the next occurrence of 6:00 AM

Example:

```
waituntil 6:00 AM, mon
```

Pauses growth until the next occurrence of 6:00 AM on a Monday.

---

```
waituntil ([low limit] < [temp loop] < [high limit]) [time in seconds]
```

Pauses until the specified temperature loop is within range for the specified time in seconds.

Example:

```
waituntil (595 < sub < 605) 60
```

Waits until the substrate temperature is between 595 and 605 continuously for 60 seconds. The timer is started when the measured temperature enters the specified range. The timer is reset if the temperature goes outside the range.

---

```
waitop (message)
```

Pauses the growth until operator clicks the "continue" button in the pop up dialogue box. The message string will be displayed in the dialogue box.

Example:

```
waitop (Wait until pyrometer reads 600C)
```

Pauses growth until operator clicks “continue” button. The message, “Wait until pyrometer reads 600C” appears in the pop up window.

---

```
layer [material] [thickness]
or
l [material] [thickness]
```

Grows a layer of the specified material and thickness. The material name must have already been defined in the shutters setup window. Note that the material names are not case sensitive. Typical materials include:

GaAs  
pGaAs  
nGaAs  
AlGaAs  
pAlGaAs  
nAlGaAs  
AlAs  
pAlAs  
nAlAs

Example:

```
layer pGaAs 120.42
```

Grows a p-doped GaAs layer 120.42 Å thick.

```
layer [material]=[user variable name]
or
l [material]=[user variable name]
```

Grows a layer of the specified material and with a thickness defined by the value stored in the variable indicated. The material name must have already been defined in the shutters setup window. Note that the material names are not case sensitive. Typical materials include:

GaAs  
pGaAs  
nGaAs  
AlGaAs  
pAlGaAs  
nAlGaAs  
AlAs

pAlAs  
nAlAs

Example:

```
eval t=497  
layer pGaAs=t
```

Grows a p-doped GaAs layer 497Å thick.

Example:

```
ask t (Please enter a value for thickness)  
layer pGaAs=t
```

Grows a p-doped GaAs layer with the thickness entered by the user.

---

```
timed [material] [time]
```

Grows a layer of the specified material and time (in seconds). This is a variation of the layer command that ignores the current rate information and uses a value of 1.

Example:

```
timed pGaAs 3600
```

Grows a p-doped GaAs layer for 3600 seconds.

---

```
open [shutter1], [shutter2], [shutter3],
```

Opens the specified shutters. These shutters will remain open until the “close” command is issued regardless of what other layers are grown. It is not recommended to use the open and close commands to grow layers. The open command is useful for opening other shutters such as a pyrometer shutter. It is also useful for opening a shutter that should stay open during the entire growth (arsenic shutter).

Example:

```
open Ga, In
```

Opens the Ga and In shutters.

---

```
close [shutter1], [shutter2], [shutter3],
```

Closes shutters that have been opened using the “open” command.

Example:

```
close Ga, In
```

Closes the Ga and In shutters.

Example 2:

```
close all
```

Closes all shutters that have been opened using the “open” command.

---

```
structure (NAME)
```

```
command 1
```

```
command 2
```

```
command 3
```

```
.
```

```
.
```

```
.
```

```
es
```

or

```
s (NAME)
```

```
command 1
```

```
command 2
```

```
command 3
```

```
.
```

```
.
```

```
.
```

```
es
```

Defines a structure that can be called anywhere throughout the growth recipe. The structure must be defined before it can be used in the recipe.

Example:

```
structure (quantum well)
```

```
layer AlGaAs 100
```

```
layer GaAs 70
```

```
layer AlGaAs 100
```

```
es
```

Defines a structure called “quantum well” that can be called later in the recipe. The end of the structure definition is indicated by the `es` command.

Example:

```
structure (multi-quantum well)
repeat 4
quantum well
er
es
```

Defines a structure called “multi-quantum well” that repeats the previously defined “quantum well” 4 times. One structure can call another structure, but only if the structure being called has already been defined.

---

```
repeat [integer]
command 1
command 2
command 3
.
.
.
er
```

Repeats the commands within the loop the specified number of times. During recipe execution, a pop-up display indicates the number of loop iterations remaining.

Example:

```
repeat 8
layer GaAs 5.0
layer AlAs 5.0
er
```

Grows a 5 Å GaAs layer followed by a 5 Å AlAs layer and repeats 8 times.

Example 2:

```
repeat 8
repeat 4
command
er
er
```

Repeats *command* 32 times. Illustrates that nested loops can be used. Each loop must be terminated separately with the `er` command.

---

```
rate [rate name] [growth rate]
OR
gr [rate name] [growth rate]
```

Sets the value for the specified growth rate.

Example:

```
rate AlAs 2.342
```

Set the AlAs growth rate to 2.342 Å/s.

```
rate [rate name]=[user variable name]
OR
gr [rate name]=[user variable name]
```

Sets the value for the growth rate based upon the value in the specified user variable.

Example:

```
eval r=2.158
rate AlAs=r
```

Set the AlAs growth rate to 2.158 Å/s.

---

```
temp [cell] [setpoint]
OR
t [cell] [setpoint]
OR
temp [cell] [setpoint] [ramprate]
OR
t [cell] [setpoint] [ramprate]
```

Sets the temperature of the specified heater to the desired setpoint. The ramp rate is an optional parameter that is used only on supported temperature loops. The units for the ramp rate are whatever the internal units of the eurotherm are set to.

Example:

```
temp ga 945
```

Set the ga cell temperature to 945 °C.

```
temp ga 1200 0.5
```

Set the ga cell temperature to 1200 °C at a ramp rate of 0.5 °C/s using the internal ramping built into the eurotherm.

---

```
temp [cell]=[user variable name]
OR
t [cell]=[user variable name]
```

By including the = sign in the temp command with a previously defined user variable, a temperature setpoint can be set to the value stored in the specified variable.

Example:

```
eval silicontemp=1224
temp si=silicontemp
```

Sets the silicon cell temperature to 1224.

---

```
setparameter [parameter name] [loop name] [value to send]
```

Sets a parameter in an external temperature controller to the specified value.

Example:

```
setparameter XP sub 25
```

Sets the substrate proportional band to 25.

```
setparameter [parameter name] [loop name]=[user variable name]
```

By including the = sign in the setparameter command with a previously defined user variable, a parameter stored in an external temperature controller can be set to the value stored in the specified user variable.

Example:

```
eval subprop=20
setparameter XP sub=substrateP
```

Sets the substrate temperature loop proportional band to 20 (the value stored in the variable name subprop).

---

```
eval [variable name] = expression
```

Sets the variable name to the value defined by expression. You may use the following operators:

+, -, \*, /, ^

and the following functions:

abs(x)	Absolute Value	Returns the absolute value of x.
acos(x)	Inverse Cosine	Computes the inverse cosine of x in radians.
acosh(x)	Inverse Hyperbolic Cosine	Computes the inverse hyperbolic cosine of x.
asin(x)	Inverse Sine	Computes the inverse sine of x in radians.
asinh(x)	Inverse Hyperbolic Sine	Computes the inverse hyperbolic sine of x.
atan(x)	Inverse Tangent	Computes the inverse tangent of x in radians.
atanh(x)	Inverse Hyperbolic Tangent	Computes the inverse hyperbolic tangent of x.
ceil(x)	Round To +Infinity	Rounds x to the next higher integer (smallest integer x).
ci(x)	Cosine Integral	Computes the cosine integral of x where x is any real number.
cos(x)	Cosine	Computes the cosine of x, where x is in radians.
cosh(x)	Hyperbolic Cosine	Computes the hyperbolic cosine of x.
cot(x)	Cotangent	Computes the cotangent of x (1/tan(x)), where x is in radians.
csc(x)	Cosecant	Computes the cosecant of x (1/sin(x)), where x is in radians.
exp(x)	Exponential	Computes the value of e raised to the x power.
expm1(x)	Exponential (Arg) – 1	Computes one less than the value of e raised to the x power ((e^x) – 1).
floor(x)	Round To –Infinity	Truncates x to the next lower integer (largest integer x).
gamma(x)	Gamma Function	(n + 1) = n! for all natural numbers n.
getexp(x)	Mantissa & Exponent	Returns the exponent of x.
getman(x)	Mantissa & Exponent	Returns the mantissa of x.
int(x)	Round To Nearest	Rounds x to the nearest integer.
intrz(x)	—	Rounds x to the nearest integer between x and zero.
ln(x)	Natural Logarithm	Computes the natural logarithm of x (to the base of e).
lnp1(x)	Natural Logarithm (Arg +1)	Computes the natural logarithm of (x + 1).
log(x)	Logarithm Base 10	Computes the logarithm of x (to the base of 10).
log2(x)	Logarithm Base 2	Computes the logarithm of x (to the base of 2).
pi(x)	Represents the value = 3.14159...	pi(x) = x * pi(1) = pi(2.4) = 2.4 *
rand( )	Random Number (0 – 1)	Produces a floating-point number between 0 and 1 exclusively.
sec(x)	Secant	Computes the secant of x, where x is in radians (1/cos(x)).
si(x)	Sine Integral	Computes the sine integral of x where x is any real number.
sign(x)	Sign	Returns 1 if x is greater than 0, returns 0 if x is equal to 0, and returns –1 if x is less than 0.
sin(x)	Sine	Computes the sine of x, where x is in radians.
sinc(x)	Sinc	Computes the sine of x divided by x (sin(x)/x), where x is in radians.
sinh(x)	Hyperbolic Sine	Computes the hyperbolic sine of x.
spike(x)	Spike Function	spike(x) returns: 1 if 0 <= x < 1 0 for any other value of x.
sqrt(x)	Square Root	Computes the square root of x.
square(x)	Square Function	square(x) returns: 1 if 2n <= x < (2n + 1) 0 if (2n + 1) <= x < (2n + 2) where x is any real number and n is any integer.
step(x)	Step Function	step(x) returns: 0 if x < 0 1 if any other condition obtains.
tan(x)	Tangent	Computes the tangent of x, where x is in radians.
tanh(x)	Hyperbolic Tangent	Computes the hyperbolic tangent of x.

**IMPORTANT NOTE:** Do not try to create any user defined variables with the same names as predefined functions.

Example:

```
eval silicontemp=1200
eval tempadjust=5
eval silicontemp=silicontemp+tempadjust
eval x=a^b
```

---

```
writefile ([filename]; arg1, arg2, arg3, ...)
OR
wfile ([filename]; arg1, arg2, arg3, ...)
```

Writes data to a text file. The filename will be preceded by the date string YYMMDD. If the file already exists, the file will be appended. Arguments can be user variables or they can be text strings. To write a text string, put it in quotes.

Example:

```
readpyro actualtemp
writefile (pyroreading; "Pyrometer reading=", actualtemp)
```

Creates a text file called "pyroreading" with the following data:

```
Pyrometer reading= 621
```

The columns are separated by tabs

---

```
ask [user variable name] (Comment for user)
```

Pops up a window and asks the user to enter a value for the specified variable. This allows changes to the growth conditions without rewriting the recipe. The string in parenthesis appears in the window when the user is prompted to enter a value.

Example:

```
ask silicontemp (Please enter desired silicon temperature)
temp si=silicontemp
```

Pops up a window and asks the user to enter a value for the variable "silicontemp". The value is then written to the silicon cell. The window will also show the message "Please enter desired silicon temperature".

---

```
fitexp (temp1, temp2, temp3, ... tempn; flux1, flux2, flux3, ...
fluxn; var1, var2)
```

Fits the data stored in the variables entered for temperature and flux and stores the results in variables named var1 and var2 (or any other names you choose). The data is fit to the following equation:

$$\text{Flux} = A \exp(-E_a/k(T+273.15))$$

The fitted value for the amplitude, A, is stored in the variable named var1. The fitted value for the activation energy (in eV) is stored in var2. Temperature values should be in degrees C.

Example:

```
fitexp (t1, t2, t3; f1, f2, f3; Amp, Ea)
```

Calculates the fitting parameters for the data stored in t1,t2,t3 and f1,f2,f3. The amplitude is stored in Amp and the activation energy is stored in Ea.

To calculate the temperature required to get a given flux, use the following line in your recipe:

```
eval t=-Ea/(8.617385E-5*ln(desiredflux/Amp))-273.15
```

---

```
writevalue [instrument name] [numeric value]
```

Writes a numeric value to a writable instrument parameter. The value is checked to make sure it is within the defined acceptable range for that instrument before it is sent.

Example:

```
writevalue rotation 10
```

Sends a value of 10 to the instrument called rotation.

---

```
writetext [instrument name] [text string]
```

Example:

```
writevalue ellipsometer start
```

Sends the string "start" to the instrument called ellipsometer.

---

```
plugin [plugin name] ([parameter 1] [value 1]; [parameter 2]  
[value 2])
```

Writes a value or values to a plugin parameter(s). Use semicolons to separate multiple commands.

Example:

```
plugin bufferbake (targettemp 250; dwelltime 1800; running true)
```

Sets the values of plugin variables targettemp, dwelltime and running to the indicated values.

```
plugin [plugin name] ([parameter 1]=[user variable name])
```

Writes the numeric value stored in the specified user variable to the indicated parameter. May be combined with other commands using a semicolon to separate commands.

Example:

```
eval buffertemptarget=270  
plugin bufferbake (targettemp=buffertemptarget; running true)
```

Sets the values of plugin variables targettemp, dwelltime and running to the indicated values.