

File @@CLASS.h

==> Header

```
class @@CLASS
{
    void action( ... );
    ==> MemberDecl
    ==> Global
}
```

File @@CLASS.cc

#include "@@CLASS.h"

@@CLASS::action( ... ) : called at each time step

```
{
    ==> Equation
}
@@CLASS::@@CLASS( void )
{
    ==> Constructor : called only once before anything else
}
@@CLASS::~@@CLASS( void )
{
    ==> Destructor
}
==> MemberImpl
```

<b>MemberDecl</b> = "double a; int i; double f (double x = 1); "
<b>Global</b> = "double LENGTH; double VEL;"
<b>Header</b> = ""
<b>Constructor</b> = ""
<b>Destructor</b> = ""
<b>Equation</b> = "out0 = PI * f(x) * a;"
<b>DoWhenGlobalChanges</b> = " a = LENGTH * VEL; "
<b>MemberImpl</b> = " double @@CLASS::f(double x) { return x * a; } "

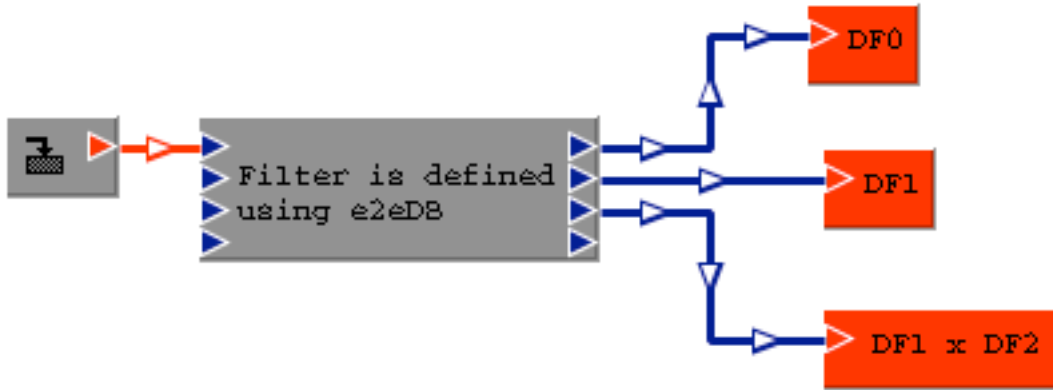
```
C0.h
class @@CLASS: public e2eSource
public:
    @@CLASS(void);
    virtual ~@@CLASS(void)
    virtual void action (funcData func_data, vector <double *> &in,
        <double> *out, <vector <double> *> &inVec,
        <vector <double> > *outVec;
    virtual bool setGlobal(void);
    @@GLOBAL
    @@MEMBER_DECL
};
```

```
C0.cc
#include <iostream>
#include <time.h>
#include <math.h>
@@HEADERS
#include "C0.h"

@@CLASS::@@CLASS ( ) : e2eSource()
{
    @@CLASS::~@@CLASS()
}
@@CLASS::action ( ..... )
{
}

bool @@CLASS::setGlobals( e2eDB &db, string &errors )
{
    bool found_errors =false;
    if (db.valueExists("LENGTH"))LENGTH = db.getVal("LENGTH");
    else { compose_error(errors, "LENGTH"); }
    // do the same for all gbbals
    @@DO_WHEN_GLOBAL_CHANGES
    return found_errors;
}
@@MEMBER_IMPL
```

A FUNC\_X example using digital filters. The filter is defined in e2eDB.mcr using e2e macro.



	Current Value	Type
Equations	<b>@@INCLUDE CC_X_main.cc</b>	<b>par</b>
MemberDecl	<b>@@INCLUDE CC_X_memberDecl.h</b>	<b>par</b>
MemberImpl	<b>DEFAULT</b>	<b>par</b>
Global	<b>@@INCLUDE CC_X_global.h</b>	<b>par</b>
Header	<b>DEFAULT</b>	<b>par</b>
Constructor	<b>@@INCLUDE CC_X_constructor.cc</b>	<b>par</b>
Destructor	<b>DEFAULT</b>	<b>par</b>
DoWhenGlobalChanges	<b>DEFAULT</b>	<b>par</b>
showInstanceSettings	<b>DEFAULT</b>	<b>par</b>
#include	<b>DEFAULT</b>	<b>par</b>

**@@INCLUDE filename** includes a file and does necessary substitution, including @@CLASS.

FUNC\_X replaces `/*@@BEGIN_COMMENT*/` by `/*` and `/*@@END_COMMENT*/` by `*/`.

The content of the following CC\_X\_main.cc is a legitimate cc file, and a language sensitive editors, like emacs, handles it appropriately. In FUNC\_X, only the blue part is used as a code.

---

### CC\_X\_main.cc : main action called at each time step

```
/*@@BEGIN_COMMENT*/
void @@CLASS::action
( vector <double *> &in, vector <double> *out,
vector <vector <double > *> &inVec, vector <vector <double> >
*outVec )
{
/*@@END_COMMENT*/

    double noise = WhiteNoiseFunc( in0 ); // member function
                                           // declared in memDecl

    // filter related functions are listed in e2eREADME.txt
    out0 = DF0.filterApply( noise );      // filters defined in
    out1 = DF1.filterApply( noise );      // constructor
    out2 = DF0Copy.filterApply( out1 );   // using macro values

/*@@BEGIN_COMMENT*/
}
/*@@END_COMMENT*/
```

---

### CC\_X\_global.h : e2e macro names need to be declared here

```
// e2e macros are all reals.
double NumZeros0;
double NumZeros1;
double NumPoles0;
double NumPoles1;

// macro names are listed one per line
double gain0;
double zero00;
double zero01;
double pole00;
double pole01;
double zQ00;
double zQ01;
```

```
double pQ00;  
double pQ01;  
  
double gain1;  
double zero10;  
double zero11;  
double pole10;  
double pole11;  
double zQ10;  
double zQ11;  
double pQ10;  
double pQ11;
```

---

**e2eDB.mcr : all macro names declated in global section need to be defined as macro names.**

```
NumZeros0 = 2  
NumZeros1 = 1  
NumPoles0 = 2  
NumPoles1 = 2
```

```
% macro names are listed one per line
```

```
gain0 = 1000  
zero00 = 0  
zero01 = 10  
pole00 = 100  
pole01 = 100  
zQ00 = 0  
zQ01 = 0  
pQ00 = 0  
pQ01 = 0  
  
gain1 = 1000  
zero10 = 10  
zero11 = 20  
pole10 = 100  
pole11 = 1000  
zQ10 = 0  
zQ11 = 0  
pQ10 = 0  
pQ11 = 0
```

---

**CC\_X\_memberDecl.h : member data and member functions declared (and defined as inline functions)**

```
// digital filters
DF DF0, DF1, DF0Copy;

// member functions
adlib_real WhiteNoiseFunc( adlib_real amp )
{
    return amp*e2eRand::rndnorm( ) / sqrt( TIME_STEP );
}

// build digital filter up to 2 zeros and poles
void buildDF
( DF* dfp, string name,
  double gain, int NumZeros, int NumPoles,
  double z0, double z1, double zQ0, double zQ1,
  double p0, double p1, double pQ0, double pQ1 )
{
    double zs[2], zQ[2], ps[2], pQ[2];

    zs[0] = z0; zs[1] = z1; zQ[0] = zQ0; zQ[1] = zQ1;
    ps[0] = p0; ps[1] = p1; pQ[0] = pQ0; pQ[1] = pQ1;

    dfp->addKZPHzQ( gain, NumZeros, zs, zQ, NumPoles, ps, pQ );
    dfp->setName( name );
}

```

---

**CC\_X\_constructor.cc : this is called once before action starts**

```
/*@@BEGIN_COMMENT*/
void @@CLASS::@@CLASS( void )
{
    /*@@END_COMMENT*/

    buildDF( &DF0, string("DF0"), gain0,
             int( NumZeros0+0.5 ), int( NumPoles0+0.5 ),
             zero00, zero01, zQ00, zQ01,
             pole00, pole01, pQ00, pQ01 );

    buildDF( &DF1, string("DF1"), gain1,

```

```
int( NumZeros1+0.5 ), int( NumPoles1+0.5 ),  
zero10, zero11, zQ10, zQ11,  
pole10, pole11, pQ10, pQ11 );
```

```
DF0Copy = DF0;  
DF0Copy.setName( "DF0Copy" );
```

```
/*@@BEGIN_COMMENT*/  
}  
/*@@END_COMMENT*/
```