```
# Example of AI record that uses the devAiRnd
record(ai, "$(user):aiRandom")
{
    field(DESC, "Random Test")
    field(DTYP, "random")
    field(INP,  "10.0")
    field(SCAN, "1 second")
}

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

# dbd file that links DTYP="random" to devAiRnd
include "base.dbd"
device(ai,CONSTANT,devAiRnd,"random")

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

/* drvRandom.h */
double drvRandom(double upper_limit);

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

/* drvRandom.c */
#include <stdlib.h>

double drvRandom(double upper_limit)
{
    return random() * upper_limit / RAND_MAX;
}

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

/* devAiRnd.c */
/* Minimal example of device support for Ai record */

#include <stddef.h>
#include <stdio.h>
#include <string.h>
#include "alarm.h"
#include "cvtTable.h"
#include "dbDefs.h"
#include "dbAccess.h"
#include "recGbl.h"
#include "recSup.h"
#include "devSup.h"
#include "link.h"
#include "aiRecord.h"
#include "drvRandom.h"
#include "epicsExport.h"

/* Almost any device needs to maintain some data:
 * Address of hardware, state of comminucation with device, ...
 * In this case it's only the upper limit of the random
 * number generation.
 */
typedef struct
{
    double upper_limit;
}   devRndData;

static long init_record(aiRecord *rec)
{
    devRndData *data;
```

```
    /* ai.inp must be a CONSTANT, defining the upper limit */
    if (rec->inp.type != CONSTANT)
    {
        recGblRecordError(S_db_badField, rec,
                          "devAiRnd (init_record) Illegal INP field");
        return S_db_badField;
    }

    data = malloc(sizeof(devRndData));
    recGblInitConstantLink(&rec->inp, DBF_DOUBLE, &data->upper_limit);

    /* device private (dpvt) is where we can park our device data */
    rec->dpvt = data;

    return 0;
}

static long read_ai(aiRecord *rec)
{
    devRndData *data = (devRndData *) rec->dpvt;
    if (data)
    {
        rec->val = drvRandom(data->upper_limit);
        rec->udf = FALSE;
    }
    return 2; /* 2 == don't convert rval to val */
}

/*Create the device support entry table  */
struct
{
    long        number;
    DEVSUPFUN   report;
    DEVSUPFUN   init;
    DEVSUPFUN   init_record;
    DEVSUPFUN   get_ioint_info;
    DEVSUPFUN   read_ai;
    DEVSUPFUN   special_linconv;
}   devAiRnd =
{
    6,
    NULL,
    NULL,
    init_record,
    NULL,
    read_ai,
    NULL
};

epicsExportAddress(dset,devAiRnd);
```