loadmemory

```
BOOL IINSTANCE::loadmemory (CHAR *filename, VOID *buffer, UINT size, UINT base = 0, UINT shift = 0)
```

Parameters:

CHAR ***filename** - имя загружаемого файла. Расширение файла должно соответствовать поддерживаемым форматам (***.BIN**, ***.HEX**, ***.S19**).

VOID ***buffer** - указатель на массив данных, куда будет загружен файл.

UINT **size** - размер загружаемых данных, в байтах. Может быть больше, чем размер реальных данных.

UINT **base** - смещение адреса относительно начала файла, в байтах, в нулевой нотации.

UINT **shift** - пропуск данных, в нулевой нотации - загружается каждый 2^{shift} байт, начиная с **base** байта.

Return Value:

BOOL - результат работы функции:

TRUE - завершилась успешно.

FALSE - завершилась с ошибкой.

Функция загружает заданный файл, в заданном формате, в заданный буфер данных, с возможностью смещения и пропуска байтов данных.

Функция поддерживает форматы: **BIN**, **HEX**, **S19**.

Пример использования функции:

```
    IINSTANCE         *Inst;            // Экземпляр интерфейса модели
    CREATEPOPUPSTRUCT *cps;             // Структура для POPUP-окон
    IMEMORYPOPUP      *PopupMemory;     // IMEMORYPOPUP-окно
    BYTE               byData[48];      // Массив данных
   ...
IDSIMPIN *PinIP0;
//-------------------------------------------------------------------------------------------------------
INT TypeModel::isdigital(CHAR *pinname)
{
  return TRUE;
}
//-------------------------------------------------------------------------------------------------------
VOID TypeModel::setup(IINSTANCE *inst, IDSIMCKT *dsim)
{
  Inst = inst;
  PinIP0 = Inst->getdsimpin("IP0", TRUE);
  cps = new CREATEPOPUPSTRUCT;                              // Новая структура для создания окна
  cps->id = 13;                                             // ID окна
  cps->type = PWT_MEMORY;                                   // Окно IMEMORYPOPUP
  cps->caption = "=== IMEMORYPOPUP ===";                    // Заголовок всплывающего окна
  cps->width = 16;                                          // Ширина всплывающего окна
  cps->height = 4;                                          // Высота всплывающего окна
  cps->flags = 0;                                           // Флаги создаваемого всплывающего окна
  PopupMemory = (IMEMORYPOPUP *)Inst->createpopup(cps);     // Создать всплывающее окно
  PopupMemory->setmemory(0x0F00, byData, 48);               // Привязать массив к окну
}
//-------------------------------------------------------------------------------------------------------
VOID TypeModel::simulate(ABSTIME time, DSIMMODES mode)
{
  if (PinIP0->isnegedge())                          // Срабатывание по отрицательному фронту LOAD
  {
    Inst->loadmemory("Test.hex", byData, 16);
    Inst->loadmemory("Test.hex", &byData[16], 16, 5);
    Inst->loadmemory("Test.hex", &byData[32], 16, 0, 2);
    Inst->log("=====>> AFTER LOAD <<========================== REALTIME = %f", realtime(time));
  }
  PopupMemory->repaint();
}
```

Test.hex

```
:0F00000054455354206C6F61646D656D6F727958
:00000001FF
```

В процессе тестирования кода со схемой проводились следующие действия:

- запуск симуляции - соответствует временной отметке **0**. В процедуре **setup** производятся настройки контактов и всплывающего окна.

- далее нажималась и отпускалась **LOAD**, что привело к загрузке данных в разные области памяти, с различными параметрами загрузки, с выводом соответствующего информационного сообщения в лог симулятора.
  Первая порция данных загружается без смещения как в массиве данных, так и в загружаемой информации.
  Вторая - загружается в массив данных со смещением на **16** байт. Входная информация имеет смещение **5** байт, т.е. загрузились данные, начиная с **5**-*го* байта (в нулевой нотации).
  Третья - загружается в массив данных со смещением на **32** байта. Входная информация смещение не имеет, но установлен пропуск 2^2, т.е. загрузился каждый **4**-*й* байт данных, начиная с **0**-*го* байта.