```
void __fastcall // вызывается для завершения выполнения инструкции на АИУ i_Proc
Finalize_Except_SET(int i_Proc) // все операци кроме SET !!!!!!!!!!!!!!!!!!!!!!
{ // устанавливаются флаги готовности у входных операндов иных инструкций, зависящих
// по входным операндам от результата выполнения данной на АИУ номер i_Proc
 char Set[_SET_LEN],
      adrResult[_ID_LEN],
      adrPredicat[_ID_LEN], // поле предиката
      str[_4096], tmp[_256];
 bool s_isPredicat, // true, если ВЫПОЛНИВШИЙСЯ оператор есть ПРЕДИКАТ
      isPredicat, // true, если ЗАВИСИМЫЙ оператор ПРЕДИКАТ
      flagNot, // true, если в ЗАВИСИМОМ операторе первый символ имени флага предиката "!" или
      flagPredicat, // true, если в имени ЗАВИСИМОГО оператора есть переменная (XXX или !XXX)
      flagPredicat_TRUE; // true, если в ЗАВИСИМОМ операторе флаг предиката true (с учётом isN
//
 if( !Regim ) // не выполнять - закончить счет ------------------------------
  return;
//
 int i_Set = Mem_Proc[i_Proc].i_Set; // на этом АИУ выполнялась инструкция номер i_Set
//
 REAL Result = Mem_Proc[i_Proc].Result; // значение результата выполненной операции
//
 strcpy( adrResult, Mem_Proc[i_Proc].adrResult ); // запомнили адрес результата выполнения инс
//
 s_isPredicat = is_Predicat( Mem_Instruction[i_Set].Set ); // true, если выполнившийся операто
////////////////////////////////////////////////////////////////////////
 Add_toData( i_Set, Mem_Proc[i_Proc].adrResult, Result ); // добавить результат выполнившейся
////////////////////////////////////////////////////////////////////////
 t_printf( "-I- %s(){1}: АИУ номер %d выполнило инструкцию #%d [%s] [%d/%d/%d тактов] -I-",
                  __FUNC__, i_Proc, i_Set, Line_Set(i_Set, 1),
                  Mem_Proc[i_Proc].tick_Start, localTick, localTick - Mem_Proc[i_Proc].tick_St
//
 Vizu_Data(); // визуализировать...
//
////////////////////////////////////////////////////////////////////////
// добавили запись в набор строк Tpr для анализа загруженности АИУ.....
 snprintf(str,sizeof(str), "%10d%10d%10d%10d%10d [%s]",
               i_Proc,
               Mem_Proc[i_Proc].tick_Start, localTick, localTick - Mem_Proc[i_Proc].tick_Start
               i_Set, Line_Set(i_Set, 1));
```

```
 mTpr->Add(str); // добавили строку
//
////////////////////////////////////////////////////////////////////////////
//
 strcpy(str, ""); // очистим str
//
//=========================================================================
 for(UI i=0; i<Really_Set; i++) // по всем инструкциям из Mem_Instruction[] ===========
  {
//
   if( i == i_Set ) // кроме только что выполненной.........................
    continue;
//
   strcpy( Set, Mem_Instruction[i].Set ); // ... так удобнее для дальнейшей работы !
   if( is_SET(Set) ) // инструкция SET уже давно едИножды выполнена !
    continue;
//
   strcpy( adrPredicat, Mem_Instruction[i].adrPredicat ); // будем работать с adrPredicat, не
//
   isPredicat = is_Predicat( Set ); // true, если это инструкция ПРЕДИКАТ
//
////////////////////////////////////////////////////////////////////////////
//
//--- проверяем, начинается ли имя переменной предиктора с '!' или '~'
   if ( !isPredicat ) // это инструкция - НЕ предиктор
    flagNot = ( adrPredicat[0] == symbolNot_1 || adrPredicat[0] == symbolNot_2 )
                ? true : false; // true, если в поле adrPredicat первый символ '!' или '~'
//
//--- проверяем, совпадает ли имя возвращённой переменной с именем ------------
//--- переменной в поле предиктора i-той инструкции ----------------------------
   flagPredicat = false;  // начальная установка
   if( !isPredicat &&  // это инструкция - НЕ предиктор
       strcmp( adrPredicat, trueLowerCase  ) && // И это НЕ статический true
       strcmp( adrPredicat, falseLowerCase ) )  // И это НЕ статический false
   {
    if( flagNot ) // имя начинается с '!' или '~'
     if( !strcmp( &adrPredicat[1],adrResult ) )  // начиная со второго символа...
      flagPredicat = true;
//
```

```
    if( !flagNot ) // имя Не начинается с '!' или '~'
     if( !strcmp( adrPredicat,adrResult ) )
      flagPredicat = true;
   }
//
//--- теперь определяем значение Result на true или false и окончательно -------
//--- (с учётом статических true/false) устанавливаем flagPredicatTrue ---------
//
   flagPredicat_TRUE = false; // начальная установка
   if( flagPredicat ) // переменная-предиктор определена, но значение ещё неизвестно
    if( ( flagNot  && !Result ) || // имя начинается с '!' или '~' и Result==false
        ( !flagNot &&  Result ) ) // имя Не начинается с '!' или '~' и Result==true
     flagPredicat_TRUE = true;
//
//--- отдельно обрабатываем статический true или false ------------------------
   if( !strcmp( adrPredicat, trueLowerCase ) ) // если true...
    flagPredicat_TRUE = true;
   if( !strcmp( adrPredicat,falseLowerCase ) ) // если false...
    flagPredicat_TRUE = false;
//
////////////////////////////////////////////////////////////////////////////////
//
   if(Mem_Instruction[i].flagExec    || // если инструкция ВЫПОЛНЯЕТСЯ "или"
      Mem_Instruction[i].flagExecOut || // уже ВЫПОЛНЕНА "или"
      Mem_Instruction[i].flagAddBuffer) // уже ДОБАВЛЕНА В БУФЕР
    continue;
//
////////////////////////////////////////////////////////////////////////////////
//
   switch( Get_CountOperandsByInstruction( Set ) ) // !!! число входных операндов инструкции S
   {
////////////////////////////////////////////////////////////////////////////////
    case 1: // ----- один операнд + (возможно) предикат ----------------------
////////////////////////////////////////////////////////////////////////////////
// ----- ВЫПОЛНИВШИЙСЯ оператор - НЕ ПРЕДИКАТ && ЗАВИСИМЫЙ оператор - НЕ ПРЕДИКАТ (1 операнд)
          if( !s_isPredicat && !isPredicat )
          {
           if( MI_adrOp1(i) ) // 1-й операнд ГОТОВ
           {
```

```
             MI_flagOp1(i) = true;
             snprintf(tmp,sizeof(tmp), " %d(1|1)", i); strcat(str, tmp); // флаг ГОТОВ у 1-го
             snprintf(tmp,sizeof(tmp), " %d(*|1)", i); strcat(str, tmp); // флаг ГОТОВ у инст
             mS->Cells[6][i+1] = Vizu_Flags(i); // визуализировали ФЛАГИ данной инструкции
            }
//
//----- если это НЕ ПРЕДИКАТНАЯ инструкция, надо ещё проверить истинность поля ПРЕДИКАТА
            if( !flagPredicat &&
                flagPredicat_TRUE ) // значение флага ПРЕДИКАТА
            {
//            Mem_Instruction[i].flagPredicat = true; // установили флаг ПРЕДИКТОР_ГОТОВ
             Mem_Instruction[i].flagPredicat_TRUE = true; // установили флаг ПРЕДИКТОР_ИСТИНЕ
             snprintf(tmp,sizeof(tmp), " %d(PredTRUE|1)", i); strcat(str, tmp); // флаг true
             mS->Cells[6][i+1] = Vizu_Flags(i); // визуализировали ФЛАГИ данной инструкции
            }
//
            if( MI_flagOp1(i) && // первый операнд ГОТОВ...
                flagPredicat_TRUE ) // ... и флаг предиката есть true
             Add_toBuffer( i ); // добавить ГКВ-команду в буфер команд для исполнения
           } // конец if( !s_isPredicat && !isPredicat )
//
// ----- ВЫПОЛНИВШИЙСЯ оператор - ПРЕДИКАТ && ЗАВИСИМЫЙ оператор - НЕ ПРЕДИКАТ (1 операнд) ...
          if( s_isPredicat && !isPredicat )
          {
           if( flagPredicat_TRUE )
            Mem_Instruction[i].flagPredicat_TRUE = true; // установим  флаг предиката
//
           if( MI_flagOp1(i) && // первый операнд ГОТОВ...
               Mem_Instruction[i].flagPredicat_TRUE ) // ... и флаг-предикат есть true
           {
            snprintf(tmp,sizeof(tmp), " %d(PredTRUE|1)", i); strcat(str, tmp); // флаг true
            Add_toBuffer( i ); // добавить ГКВ-команду в буфер команд для исполнения
           }
//
          } // конец if( s_isPredicat && !isPredicat )
//
// ----- ВЫПОЛНИВШИЙСЯ оператор - НЕ ПРЕДИКАТ && ЗАВИСИМЫЙ оператор - НЕ ПРЕДИКАТ (1 операнд)
          if( !s_isPredicat && !isPredicat )
          {
```

```
//             if( MI_adrOp1(i) ) // 1-й операнд ГОТОВ
//             {
//              Mem_Instruction[i].flagOp1 = true;
//              snprintf(tmp,sizeof(tmp), " %d(1|1)", i); strcat(str, tmp); // флаг ГОТОВ у 1-
//              mS->Cells[6][i+1] = Vizu_Flags(i); // визуализировали ФЛАГИ данной инструкции
//             }
//----- если эта инструкция-НЕ ПРЕДИКАТ, надо проверить флаг (и его значение) поля предиката
            if( !flagPredicat && // это инструкция НЕ ПРЕДИКАТ
                flagPredicat_TRUE ) // значение флага ПРЕДИКАТА
             {
              Mem_Instruction[i].flagPredicat_TRUE = true; // установили флаг ПРЕДИКАТ_ИСТИНЕН
              snprintf(tmp,sizeof(tmp), " %d(PredTRUE|1)", i);
              strcat(str, tmp); // флаг true предиката
              mS->Cells[6][i+1] = Vizu_Flags(i); // визуализировали ФЛАГИ данной инструкции
             }
//
             if( MI_flagOp1(i) && // 1-й операнд ГОТОВ...
                 Mem_Instruction[i].flagPredicat_TRUE ) // ... и флаг предиката есть true
              Add_toBuffer( i ); // добавить ГКВ-команду в буфер команд для исполнения
//
            } // конец if( !s_isPredicat && !isPredicat )
//
// ----- ВЫПОЛНИВШИЙСЯ оператор - ПРЕДИКАТ && ЗАВИСИМЫЙ оператор - ПРЕДИКАТ (1 операнд) ...
            if( s_isPredicat && isPredicat )
            {
             if( MI_adrOp1(i) ) // 1-й операнд ГОТОВ
             {
              MI_flagOp1(i) = true;
              snprintf(tmp,sizeof(tmp), " %d(Pred 1|1)", i); strcat(str, tmp); // флаг ГОТОВ у
              snprintf(tmp,sizeof(tmp), " %d(Pred *|1)", i); strcat(str, tmp); // флаг ГОТОВ у
              mS->Cells[6][i+1] = Vizu_Flags(i); // визуализировали ФЛАГИ данной инструкции
             }
//
             if( MI_flagOp1(i) ) // 1-й операнд ГОТОВ...
              Add_toBuffer( i ); // добавить ГКВ-команду в буфер команд для исполнения
//
            }  // конец if( s_isPredicat && isPredicat )
//
////////////////////////////////////////////////////////////////////////////
```

```
            break; // конец обработки оператора с одним операндом + возможно, предиктор
////////////////////////////////////////////////////////////////////////////////
    case 2: // ----- два операнда + (возможно) предикат ----------------------
////////////////////////////////////////////////////////////////////////////////
// ----- ВЫПОЛНИВШИЙСЯ оператор - НЕ ПРЕДИКАТ && ЗАВИСИМЫЙ оператор - НЕ ПРЕДИКАТ (2 операнда)
            if( !s_isPredicat && !isPredicat )
            {
             if( MI_adrOp1(i) ) // 1-й операнд ГОТОВ
             {
              MI_flagOp1(i) = true;
              snprintf(tmp,sizeof(tmp), " %d(1|2)", i); strcat(str, tmp); // флаг ГОТОВ у 1-го
//
              if( MI_flagOp1(i) &&
                  MI_flagOp2(i) )
               { snprintf(tmp,sizeof(tmp), " %d(*|2)", i); strcat(str, tmp); } // флаг ГОТОВ у
              mS->Cells[6][i+1] = Vizu_Flags(i); // визуализировали ФЛАГИ данной инструкции
             }
//
             if( MI_adrOp2(i) ) // 2-й операнд ГОТОВ
             {
              MI_flagOp2(i) = true;
              snprintf(tmp,sizeof(tmp), " %d(2|2)", i); strcat(str, tmp); // флаг ГОТОВ у 2-го
//
              if( MI_flagOp1(i) &&
                  MI_flagOp2(i) )
               { snprintf(tmp,sizeof(tmp), " %d(*|2)", i); strcat(str, tmp); } // флаг ГОТОВ у
              mS->Cells[6][i+1] = Vizu_Flags(i); // визуализировали ФЛАГИ данной инструкции
             }
//
//----- если эта инструкция-НЕ ПРЕДИКАТ, надо проверить флаг (и его значение) поля предиката
             if( !flagPredicat && // это инструкция НЕ ПРЕДИКАТ
                 flagPredicat_TRUE ) // значение флага ПРЕДИКАТА
             {
              Mem_Instruction[i].flagPredicat_TRUE = true; // установили флаг ПРЕДИКАТ_ИСТИНЕН
              snprintf(tmp,sizeof(tmp), " %d(PredTRUE|2)", i); strcat(str, tmp); // флаг true
              mS->Cells[6][i+1] = Vizu_Flags(i); // визуализировали ФЛАГИ данной инструкции
             }
//
             if( MI_flagOp1(i) && // первый операнд ГОТОВ...
```

```
                MI_flagOp2(i) && // второй операнд ГОТОВ...
                Mem_Instruction[i].flagPredicat_TRUE ) // ... и флаг предиката есть true
              Add_toBuffer( i ); // добавить ГКВ-команду в буфер команд для исполнения
//
            } // конец if( !s_isPredicat && !isPredicat )
//
// ----- ВЫПОЛНИВШИЙСЯ оператор - НЕ ПРЕДИКАТ && ЗАВИСИМЫЙ оператор - ПРЕДИКАТ (2 операнда) ..
            if( !s_isPredicat && isPredicat )
            {
             if( MI_adrOp1(i) ) // 1-й операнд ГОТОВ
             {
              MI_flagOp1(i) = true;
              snprintf(tmp,sizeof(tmp), " %d(1|1)", i); strcat(str, tmp); // флаг ГОТОВ у 1-го
              mS->Cells[6][i+1] = Vizu_Flags(i); // визуализировали ФЛАГИ данной инструкции
             }
//
             if( MI_adrOp2(i) ) // 2-й операнд ГОТОВ
             {
              MI_flagOp2(i) = true;
              snprintf(tmp,sizeof(tmp), " %d(2|2)", i); strcat(str, tmp); // флаг ГОТОВ у 2-го
//
              if( MI_flagOp1(i) &&
                  MI_flagOp2(i) )
               { snprintf(tmp,sizeof(tmp), " %d(*|2)", i); strcat(str, tmp); } // флаг ГОТОВ у
              mS->Cells[6][i+1] = Vizu_Flags(i); // визуализировали ФЛАГИ данной инструкции
             }
//
             if( MI_flagOp1(i) && // первый операнд предиката ГОТОВ... у предиката нет предика
                 MI_flagOp1(i) )  // второй операнд ГОТОВ...
              Add_toBuffer( i ); // добавить ГКВ-команду в буфер команд для исполнения
            } // конец if( !s_isPredicat && isPredicat )
//
// ----- ВЫПОЛНИВШИЙСЯ оператор - ПРЕДИКАТ && ЗАВИСИМЫЙ оператор - НЕ ПРЕДИКАТ (2 операнда) ..
            if( s_isPredicat && !isPredicat )
            {
             if( flagPredicat_TRUE )
              Mem_Instruction[i].flagPredicat_TRUE = true; // установим  флаг предиката
//
             if( MI_flagOp1(i) && // первый операнд ГОТОВ...
```

```
                MI_flagOp2(i) && // второй операнд ГОТОВ...
                Mem_Instruction[i].flagPredicat_TRUE ) // ... и флаг предиката есть true
             {
              snprintf(tmp,sizeof(tmp), " %d(PredTRUE|2)", i); strcat(str, tmp); // флаг true
              Add_toBuffer( i ); // добавить ГКВ-команду в буфер команд для исполнения
             }
//
            } // конец if( s_isPredicat && !isPredicat )
//
// ----- ВЫПОЛНИВШИЙСЯ оператор - не ПРЕДИКАТ && ЗАВИСИМЫЙ оператор - ПРЕДИКАТ (2 операнда) ..
            if( !s_isPredicat && isPredicat )
            {
             if( MI_adrOp1(i) ) // 1-й операнд ГОТОВ
             {
              MI_flagOp1(i) = true;
              snprintf(tmp,sizeof(tmp), " %d(1|2)", i); strcat(str, tmp); // флаг ГОТОВ у 1-го
//
              if( MI_flagOp1(i) &&
                  MI_flagOp2(i) )
               { snprintf(tmp,sizeof(tmp), " %d(*|2)", i); strcat(str, tmp); } // флаг ГОТОВ у
             mS->Cells[6][i+1] = Vizu_Flags(i); // визуализировали ФЛАГИ данной инструкции
             }
//
             if( MI_adrOp2(i) ) // 2-й операнд ГОТОВ
             {
              MI_flagOp2(i) = true;
              snprintf(tmp,sizeof(tmp), " %d(2|2)", i); strcat(str, tmp); // флаг ГОТОВ у 2-го
//
              if( MI_flagOp1(i) &&
                  MI_flagOp2(i) )
               { snprintf(tmp,sizeof(tmp), " %d(*|2)", i); strcat(str, tmp); } // флаг ГОТОВ у
             mS->Cells[6][i+1] = Vizu_Flags(i); // визуализировали ФЛАГИ данной инструкции
             }
//
             if( MI_flagOp1(i) && // первый операнд ГОТОВ...
                 MI_flagOp2(i) )  // второй операнд ГОТОВ...
              Add_toBuffer( i ); // добавить ГКВ-команду в буфер команд для исполнения
//
            } // конец if( !s_isPredicat && isPredicat )
```

```
//
// ----- ВЫПОЛНИВШИЙСЯ оператор - ПРЕДИКАТ && ЗАВИСИМЫЙ оператор - ПРЕДИКАТ (2 операнда) ...
            if( s_isPredicat && isPredicat )
            {
             if( MI_adrOp1(i) ) // 1-й операнд ГОТОВ
             {
              MI_flagOp1(i) = true;
              snprintf(tmp,sizeof(tmp), " %d(Pred 1|2)", i); strcat(str, tmp); // флаг ГОТОВ у
              mS->Cells[6][i+1] = Vizu_Flags(i); // визуализировали ФЛАГИ данной инструкции
             }
//
             if( MI_adrOp2(i) ) // 2-й операнд ГОТОВ
             {
              MI_flagOp2(i) = true;
              snprintf(tmp,sizeof(tmp), " %d(Pred 2|2)", i); strcat(str, tmp); // флаг ГОТОВ у
              mS->Cells[6][i+1] = Vizu_Flags(i); // визуализировали ФЛАГИ данной инструкции
             }
//
             if( MI_flagOp1(i) && // 1-й операнд ГОТОВ...
                 MI_flagOp2(i) ) // и 2-й операнд ГОТОВ!
             {
              snprintf(tmp,sizeof(tmp), " %d(Pred *|2)", i); strcat(str, tmp); // готов весь о
              Add_toBuffer( i ); // добавить ГКВ-команду в буфер команд для исполнения
             }
//
            }  // конец if( s_isPredicat && isPredicat )
//
////////////////////////////////////////////////////////////////////////
            break; // конец обработки оператора с двумя операндами + возможно, предикат
////////////////////////////////////////////////////////////////////////
   default: break; // кроме 1 или 2 операнда у оператора...
//
   } // конец switch по числу операндов у i-того оператора --------------------
//
////////////////////////////////////////////////////////////////////////
  } // конец цикла по пулу инструкций ..........................................
////////////////////////////////////////////////////////////////////////
//
////////////////////////////////////////////////////////////////////////
```

```
//
 if(strlen(str)) // если в str что-то записывалось...
  t_printf( "-I- %s(){2}: по выполнению инструкции #%d/%d установлены флаги готовности операнд
                  __FUNC__, i_Set, i_Proc, str);
//
////////////////////////////////////////////////////////////////////////
// устанавливаем флаг однократного выполнения инструкции .....................
 Mem_Instruction[i_Set].flagExecOut = true; // установили флаг ИНСТРУКЦИЯ_ИСПОЛЬЗОВАНА
// снимаем флаг ИНСТРУКЦИЯ_ВЫПОЛНЯЕТСЯ
 Mem_Instruction[i_Set].flagExec    = false; // сняли флаг ИНСТРУКЦИЯ_ВЫПОЛНЯЕТСЯ
////////////////////////////////////////////////////////////////////////
 mS->Cells[6][i_Set+1] = Vizu_Flags(i_Set); // визуализировать...
////////////////////////////////////////////////////////////////////////
 Draw_ReadyOperands(); // выделение ячеек цветом
////////////////////////////////////////////////////////////////////////
//
 Mem_Proc[i_Proc].Busy = false; // АИУ номер i_Proc теперь СВОБОДНО !!!
//
 t_printf( "-I- %s(){3}: АИУ номер %d освобождено (%s) после выполнения инструкции #%d -I-",
                  __FUNC__, i_Proc, Get_Time_asLine(), i_Set);
//
 Free_Proc ++ ; // число свободных АИУ увеличили на 1 ==========================
//
////////////////////////////////////////////////////////////////////////
 sleep_for_vizu_buffer // ждем-с для визуализации буфера
////////////////////////////////////////////////////////////////////////
 AttemptExecMaxInstructions_fromBuffer(); // пытаемся выполнить как можно больше ГКВ-инструкци
// для фактического выполнения инструкций из AttemptExecMaxInstructions_fromBuffer()
// вызывается ExecuteInstructions_ExceptET( i_Set )
////////////////////////////////////////////////////////////////////////
 sleep_for_vizu_buffer // визуализируем буфер...
////////////////////////////////////////////////////////////////////////
//
 Already_Exec ++ ; // число уже испОлненных инструкций
//
 Vizu_Flow_Exec(); // визуализация процента исполненных инструкций
//
} // --- конец Finalize_Except_SET ----------------------------------------
```