От ведущего инженера АСУ ТП
Семёнова Артемия Александровича

ОБЪЯСНИТЕЛЬНАЯ

АННОТАЦИЯ

Данный документ описывает причины останова выполнения программы ПЛК 992CS100 АСУТП ЗИФ «Павлик» 03.08.2021.

Все ниже описываемые работы выполнял ведущий инженер АСУ ТП Семёнов А А с согласования начальника службы АСУ ТП Доровика Р В.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

ПЛК — программируемый логический контроллер;

OB — организационные блоки для выполнения пользовательских программ;

FB, FC — блоки содержащие пользовательские программы.

ОПИСАНИЕ СЛУЧИВШЕГОСЯ

ПЛК 992CS100 (SIEMENS S7-400 CPU416-3DP) АСУТП ЗИФ «Павлик» управляет оборудованием и техпроцессами: сорбции; ADR; «обезврежки»; насосами хвостов сорбции; насосами гидро-подпора 065PU887U01, 065PU888U01; сгустителями 080TN450, 080TN590.

В конфигурации ПЛК прописано множество сетевых устройств PROFIBUS, которые в данный момент отключены (обесточены). Это могут быть агрегаты на обслуживании (ремонт насоса), так и резерв заложенный на подключение нового оборудования. ПЛК опрашивает данные устройства, и не получив ответ, пишет сообщение об ошибке в диагностический буфер (смотрите рисунок 1 в приложении).

Любое изменение конфигурации требует останова ПЛК и управляемого им оборудования. Поэтому подключить новые агрегаты по требованию возможно только если заложен резерв. По этой же причине невозможно отключить в конфигурации временно выведенное из работы оборудование не остановив половину технологии.

Большое кол-во недоступных по сети устройств заполняют весь буфер сообщениями, вытесняя записи о других событиях, что усложняет оперативную диагностику. Размер буфера ПЛК 120 записей.

Для решения проблемы была разработана программа ПЛК «DBUF_READ» (FB111), копирующая отфильтрованные события из диагностического буфера в таблицу в памяти (DB блок).

Программа «DBUF_READ» протестирована на стендовом ПЛК, ошибок не выявлено. 02.08.2021 программа запущена на рабочем ПЛК 992CS100, с вызовом из организационного блока OB1, выполняемого циклически. За сутки работы в отфильтрованную таблицу не было занесено ни одной записи. Но так как ПЛК несёт большое кол-во периферии, то за этот период должны случиться какие либо события, из чего сделан вывод, что в буфер обмена попадают сообщения только об отсутствующих PROFIBUS устройствами, а остальным не хватает в нём места. Тогда была выдвинута идея, что если вызывать программу сразу после возникновения событий, то их еще не успеют затереть сообщения о выключенном оборудовании. Для обработки аварий в ПЛК SIEMENS S7-400 предусмотрены OB80-89 запускаемые после обнаружения неисправностей. Вызов программы решено было добавить в данные блоки. Вызовы из OB80-89 тестировались на стендовом ПЛК, аварийных остановок не случилось. При добавлении вызова в OB85 рабочего ПЛК 992CS100, после прогрузки в 15:29:55, произошёл немедленный останов контроллера. Так как действия приведшие к этому были ясны, вызов программы из всех OB удалён, ПЛК запущен через Warm Restart.

ПРИЧИНЫ ОСТАНОВА

Анализ показал, что останов произошёл из-за переполнения local stack (L) памяти. Данная память используется для временных переменных OB, FB, FC (описанные в разделе «TEMP» блока) и для внутренних нужд вызываемых блоков. Для OB1, из которого вызывается большая часть программ, размер данной памяти 1 килобайт, для OB обработчиков аварий (OB80-89) 250 байт (смотрите рисунок 3 в приложении). Программа «DBUF_READ», для сохранения 30 записей из диагностического буфера, использует массив размером 600 байт (30 записей по 20 байт каждая), +130 байт для прочих нужд, итого 730байт. Пока FB111 вызывался из OB1, его 1024 байт L памяти было достаточно, но как только вызов программы был добавлен в OB85 с 250 байтами L памяти, произошла ошибка её переполнения (смотрите рисунок 2 приложения). На стендовом ПЛК данная проблема не была обнаружена так как OB 80-89 вызываются по прерыванию на аварии, которые за время отладки не случились. OB85 в рабочем ПЛК запускается как реакция на отсутствующие в сети устройства.

ВЫВОД

Для избежания данной ситуации:

- при обработке заметных объёмов данных использовать только память DB;
- более тщательно тестировать код перед внедрением в рабочий ПЛК.

ПРИЛОЖЕНИЕ

СООБЩЕНИЯ В ДИАГНОСТИЧЕСКОМ БУФЕРЕ

НЕТ ПОДКЛЮЧЕНИЯ К УСТРОЙСТВАМ ПО СЕТИ PROFIBUS

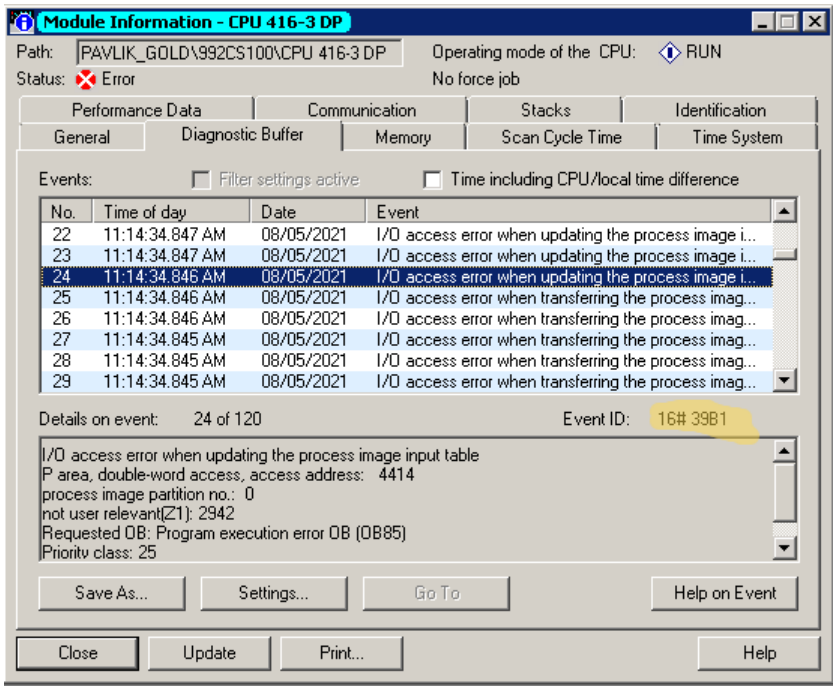

Рисунок 1: Нет подключения к устройствам по сети PROFIBUS.

ПЕРЕПОЛНЕНИЕ ПАМЯТИ LOCAL STACK

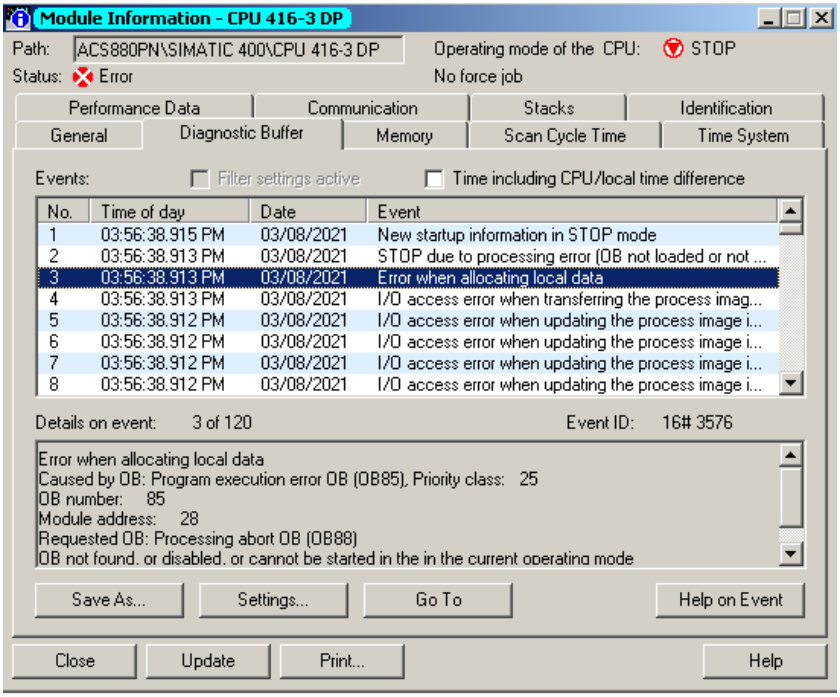

Рисунок 2: Переполнение памяти Local Stack (снимок с тестового ПЛК).

КОНФИГУРАЦИЯ ПЛК, ВЫДЕЛЕНИЕ L ПАМЯТИ OB

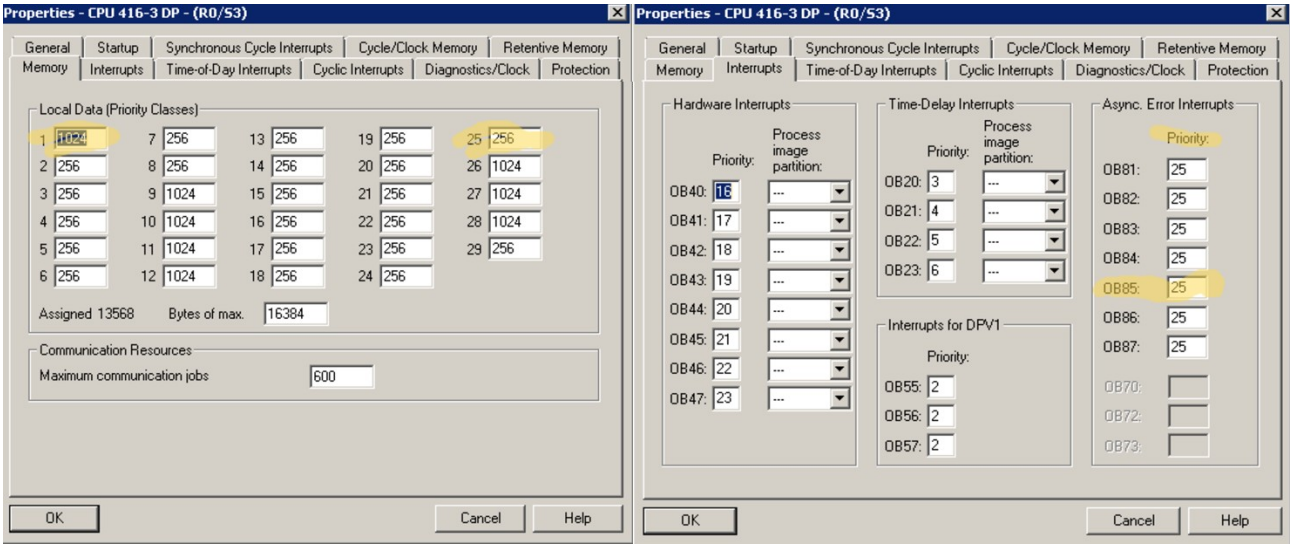

Рисунок 3: Конфигурация ПЛК, выделение L памяти OB.

РАСПРЕДЕЛЕНИЕ LOCAL STACK ПАМЯТИ

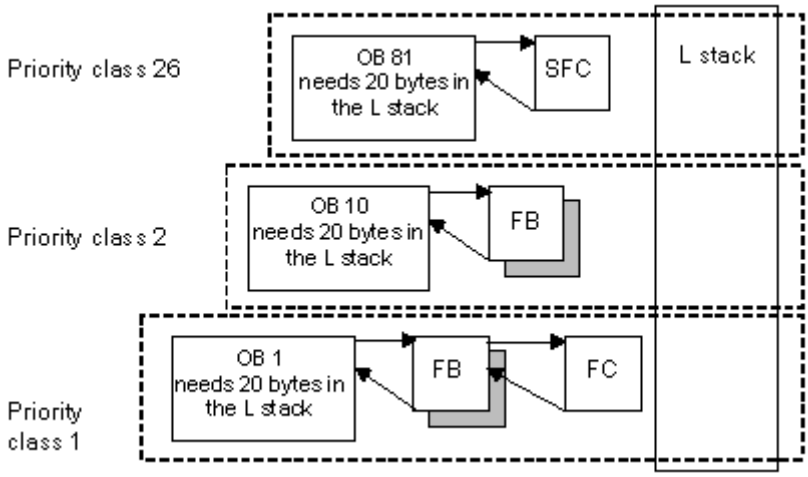

Рисунок 4: Распределение Local Stack памяти.

```
1
2   Программа DBUF_READ
3
4   FUNCTION_BLOCK FB 111
5   TITLE =Read Diagnostic Buffer CPU and Save Events to Table
6   //FB читает события из Diagnostic Buffer, исключает не нужные записи и сохраняет
7   //в таблицу EVENS DB FB
8   //Семёнов А. А.
9   AUTHOR : ART
10  VERSION : 0.1
11
12
13  VAR
14    EVENTS : ARRAY  [0 .. 30 ] OF //Таблица с очищенными сообщениями из Diag Buffer
15    STRUCT
16     ID : WORD ;
17     Priority : BYTE ;
18     OB_Number : BYTE ;
19     BlockType : BYTE ;
20     MemArea : BYTE ;
21     SourceOfError : INT ;
22     BlockNo : INT ;
23     ErrorAddr : WORD ;
24     TimeStamp : DATE_AND_TIME ;
25    END_STRUCT ;
26    CURSOR : INT ;    //Позиция курсора по таблице EVENTS
27  END_VAR
28  VAR_TEMP
29    l_evnts : ARRAY  [0 .. 30 ] OF //Таблица с EVENTами из DIAG BUFFER
30    STRUCT
31     ID : WORD ;
32     Priority : BYTE ;
33     OB_Number : BYTE ;
34     BlockType : BYTE ;
35     MemArea : BYTE ;
36     SourceOfError : INT ;
37     BlockNo : INT ;
38     ErrorAddr : WORD ;
39     TimeStamp : DATE_AND_TIME ;
40    END_STRUCT ;
41    l_evnt : STRUCT   //Просматриваемое на исключения событие
42     ID : WORD ;
43     Priority : BYTE ;
44     OB_Number : BYTE ;
45     BlockType : BYTE ;
46     MemArea : BYTE ;
47     SourceOfError : INT ;
48     BlockNo : INT ;
49     ErrorAddr : WORD ;
50     TimeStamp : DATE_AND_TIME ;
51    END_STRUCT ;
52    szl_header : STRUCT
53     LENGTHDR : WORD ;
54     N_DR : WORD ;
55    END_STRUCT ;
56    sf51_ret_val : INT ;  //+++ SFC51 - Simatic system Function Return Value >< 0 -> Error
57    sfc20_ret_val : INT ;
58    db_nr : INT ; //Вычисленный номер DB FB
59    loop_cnt : INT ;  //Счётчик итераций цикла
60    l_cursor : INT ;  //Позиция для записи в таблице с EVENTами
61    EVENT_SIZE : INT ;    //Размер в байтах записи события в таблице
62    DBUF_EVENTS_MAX_NUM : INT ;   //Макс кол-во элементов прочитахных из Diag Buffer
63    EVENTS_TBL_SIZE : INT ;   //Размер таблицы с очищенными EVENTами
64    EXCEPT_EVNTS : ARRAY  [0 .. 4 ] OF //Hex ID EVENTов исключаемых из таблицы
65    WORD ;
66    t_src_any : ANY ; //Указатель на EVNT в грязной таблице
67    t_dst_any : ANY ; //Указатель на EVNT в чистой таблице
68    l_addr : DWORD ;
69    temp0 : WORD ;
```

```
  temp1 : WORD ;
  AON : BOOL ;
  AOFF : BOOL ;
  sfc51_busy : BOOL ;
END_VAR
BEGIN
NETWORK
TITLE =
//
//
//
//
//
//


NETWORK
TITLE =Инициализация констант
//--  Размер записи 1го EVENTa в таблице в байтах
      L     20;
      T     #EVENT_SIZE;

//--  Максимальное кол-во EVENTов прочитанных из Diag Buffer
      L     20;
      T     #DBUF_EVENTS_MAX_NUM;

//--  Размер таблицы с сохранёнными EVENTами
      L     20;
      T     #EVENTS_TBL_SIZE;

//--  Перечень Event Id исключаемых из таблицы
      L     W#16#39B1;
      T     #EXCEPT_EVNTS[0];
      L     W#16#39B2;
      T     #EXCEPT_EVNTS[1];
      L     W#16#0;
      T     #EXCEPT_EVNTS[2];
      L     W#16#0;
      T     #EXCEPT_EVNTS[3];
      L     W#16#0;
      T     #EXCEPT_EVNTS[4];

NETWORK
TITLE =

      O     #AON;
      ON    #AON;
      =     L    699.0;
      A     L    699.0;
      BLD   102;
      =     #AON;
      AN    L    699.0;
      =     #AOFF;
NETWORK
TITLE =Чтение Diagnostic Buffer CPU
//EVENTы из DIAG Buffer сохраняются в локальный массив #t_evnts
      A     #AON;
      =     L    699.0;
      BLD   103;
      CALL SFC   51 (
           REQ                      := L    699.0,
           SZL_ID                   := W#16#1A0,
           INDEX                    := W#16#14,
           RET_VAL                  := #sf51_ret_val,
           BUSY                     := #sfc51_busy,
           SZL_HEADER               := #szl_header,
           DR                       := #l_evnts);
      NOP   0;
NETWORK
```

```
TITLE =

      A(    ;
      L     #szl_header.LENGTHDR;
      T     #temp0;
      SET   ;
      SAVE  ;
      CLR   ;
      A     BR;
      )     ;
      JNB   _001;
      L     #szl_header.N_DR;
      T     #temp1;
_001: NOP   0;
NETWORK
TITLE =Открытие DB FB

      L     DINO; // Номер DB FB
      T     #db_nr;
      OPN   DB [#db_nr];

      L     #CURSOR;
      T     #l_cursor;
NETWORK
TITLE =Создание указателей на таблицы
// Собираем указатели на таблицы l_evnts и EVNTS_CLR
      LAR1  P##t_src_any;
      LAR2  P##t_dst_any;
      L     B#16#10;
      T     B [AR1,P#0.0];
      T     B [AR2,P#0.0];
      L     B#16#2; // Type BYTE
      T     B [AR1,P#1.0];
      T     B [AR2,P#1.0];
      L     #EVENTS_TBL_SIZE; //Event Len 20 Byte
      T     W [AR1,P#2.0];
      T     W [AR2,P#2.0];
      L     #db_nr;
      T     W [AR1,P#4.0];
      T     W [AR2,P#4.0];
//    L     0
//    SLD   3
      L     P##l_evnts;
      T     D [AR1,P#6.0];
//    L     B#16#84                     //mem DB      L     B#16#84
//    L     B#16#87
//    T     B [AR1,P#6.0]
//--
//    L     0
//    SLD   3

      L     P##EVENTS;
      T     D [AR2,P#6.0];
//    L     B#16#84                     //mem DB
//    T     B [AR2,P#6.0]


NETWORK
TITLE =
//Прочитанные из Diag Buffer EVENTы в цикле LOOP просматриваются в массиве
//t_evnts и если запись не содержит исключения, то копируется в таблицу
//EVENTS
//на позицию указанную курсором l_cursor, после чего курсор смещается на +1
//запись.
//  Инициализация цикла на 20 элементов
      L     #DBUF_EVENTS_MAX_NUM;
nxt:  T     #loop_cnt;

//--  Вычисление адреса EVENTA в таблице
```

```
208         L     #DBUF_EVENTS_MAX_NUM;
209         L     #loop_cnt;
210         -I    ;
211         L     #EVENT_SIZE; // Размер EVENTA 20 byte
212         *I    ;
213         SLD   3;
214         T     D [AR1,P#6.0];
215         L     B#16#87; // Тип памяти (h84 - DB; h87 - LD)
216         T     B [AR1,P#6.0];
217
218
219   //--  Копируем событие в T память для анализа
220         CALL SFC   20 (
221              SRCBLK                   := #t_src_any,
222              RET_VAL                  := #sfc20_ret_val,
223              DSTBLK                   := #l_evnt);
224
225   //--  Проверяем на исключения
226         O(    ;
227         L     #l_evnt.ID;
228         L     #EXCEPT_EVNTS[0];
229         ==I   ;
230         )     ;
231         O(    ;
232         L     #l_evnt.ID;
233         L     #EXCEPT_EVNTS[1];
234         ==I   ;
235         )     ;
236         O(    ;
237         L     #l_evnt.ID;
238         L     #EXCEPT_EVNTS[2];
239         ==I   ;
240         )     ;
241         O(    ;
242         L     #l_evnt.ID;
243         L     #EXCEPT_EVNTS[3];
244         ==I   ;
245         )     ;
246         O(    ;
247         L     #l_evnt.ID;
248         L     #EXCEPT_EVNTS[4];
249         ==I   ;
250         )     ;
251
252         JCB   IADD;
253
254   //--  Вычисление  адреса позиции в таблице EVENTS
255         L     #l_cursor;
256         L     #EVENT_SIZE; //Размер EVENTa в байтах
257         *I    ;
258         L     0; // Таблица EVNEES начинается с 0 байта в DB
259         +I    ;
260         SLD   3;
261         T     D [AR2,P#6.0];
262         L     B#16#84; // Тип памяти (h84 - DB; h87 - LD)
263         T     B [AR2,P#6.0];
264
265   //--  Копирование EVENT'a
266         CALL SFC   20 (
267              SRCBLK                   := #t_src_any,
268              RET_VAL                  := #sfc20_ret_val,
269              DSTBLK                   := #t_dst_any);
270
271   //--  Смещение курсора на +1
272         L     #l_cursor;
273         L     1;
274         +I    ;
275         T     #l_cursor;
276
```

```
//--  Если позиция курсора > 20 то сбрасываем его в 0
      L     #l_cursor;
      L     #EVENTS_TBL_SIZE;
      >I    ;
      JNB   IADD;
      L     0;
      T     #l_cursor;


IADD: NOP   0;

      L     #loop_cnt;
      LOOP  nxt;
      NOP   0;

NETWORK
TITLE =Сохранение позиции курсора в DB
      L     P##CURSOR; //Указатель на курсор в DB
      T     #l_addr;
      L     #l_cursor;
      T     DBW [#l_addr];
END_FUNCTION_BLOCK
```