СТРУКТУРНОЕ ПРОГРАММИРОВАНИЕ

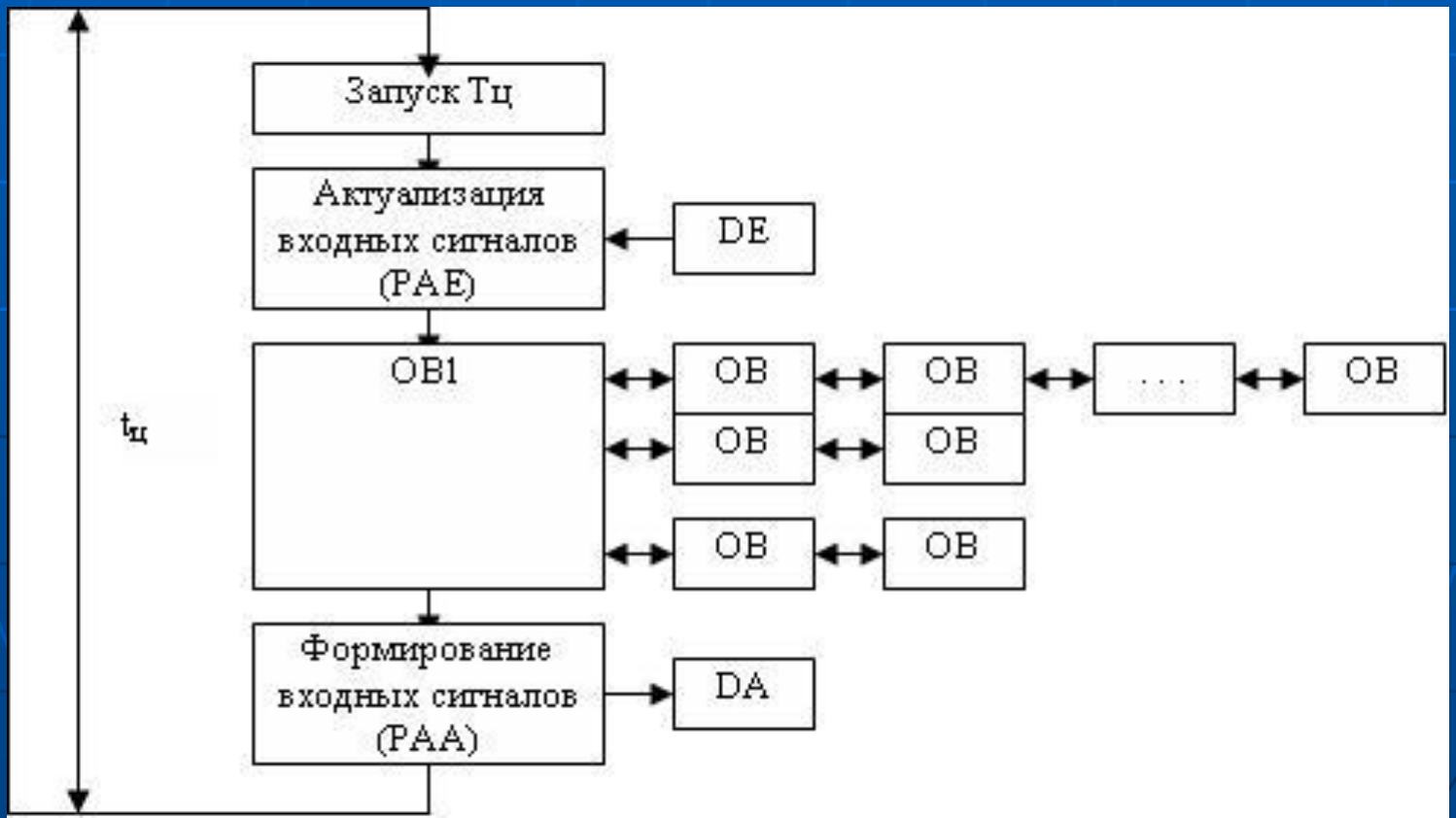
В STEP7

Обработка управляющих программ в SIMATIC S7

- Существуют следующие режимы обработки управляющих программ в SIMATIC S7:
  - Циклический;
  - Прерывания по времени;
  - Прерывания с задержкой;
  - Периодические прерывания;
  - Аппаратные прерывания;
  - Однократный запуск;
  - Обработка ошибок.
- Все эти режимы реализуются операционной системой контроллера. Для того чтобы соответствующий режим обрабатывался, в памяти контроллера должен иметься соответствующий организационный блок.

Циклический режим

- Имеет приоритет 1 (самый низший). Начинается с выполнения организационного блока OB1. Имеет следующую структуру



Циклический режим

- Длительность цикла tц есть время между началом и окончанием одного цикла. Максимальное время цикла ограничивается специальной константой (по умолчанию 100 мс). Для того, чтобы контролировать длительность цикла, в начале каждого этапа запускается специальный таймер Тц. Превышение длительностью цикла максимального значения переводит контроллер в состояние STOP.
- После запуска таймера Тц ОС контроллера опрашивает все модули входных дискретных сигналов DE и записывает их значения в специальную область отображения входов PAE.
-

Циклический режим

- Поскольку обращение к модулю DE занимает порядка 100 мс, то копирование значений входов в память сокращает время работы управляющей программы, ведь время считывания переменных из области памяти не превышает 1 мс. Изменение значений входов будет зафиксировано лишь при следующем цикле выполнения, что соответствует устранению дребезга контактов.
- Далее вызывается блок OB1, из него могут вызываться другие блоки, организующие некое дерево глубиной до 16 вызовов. Вызовы могут быть условными и безусловными.

Циклический режим

- Когда блок OB1 обработается, все выходные сигналы будут записаны в специальную область памяти – область отображения выходных сигналов (PAA). Управление передается ОС контроллера, которая переписывает значения выходных сигналов из PAA в модуль дискретных выходов DA.
- После этого происходит возврат к выполнению нового цикла.

Режимы обработки прерываний

- Перед тем, как обсуждать режимы обработки управляющих программ по прерываниям, отметим, что прерывания могут обрабатываться двумя способами:
  - на границе блоков (при вызовах других блоков). Если блоки достаточно велики и для их обработки требуется много времени, то возникнут большие задержки перед обработкой прерывания. Эта проблема решается дроблением программы на более мелкие блоки.
  - по окончанию команд (самый быстрый способ обработки прерываний). Как только контроллер обработает текущую команду языка STEP7, он перейдет к обработке прерывания.

Режимы обработки прерываний

- Если одновременно возникает несколько прерываний, то они обрабатываются в соответствии с их приоритетами.
- Обработка прерываний повышает длительность цикла обработки управляющей программы. Для уменьшения этого времени можно производить блокирование (разблокирование) любого прерывания путем вызова специфических функциональных блоков.

Прерывания по времени

- Имеют приоритет 2, прерывают циклический режим, но могут быть прерваны другими прерываниями. Предполагается, что этот вид прерываний будет обслуживаться достаточно редко: однократно по заданной дате и времени или периодически после наступления заданной даты и времени с определенным интервалом. Например, если задана дата 31.01.13 с интервалом в 1 месяц, то прерывания возникнут в январе, марте и т.д.
- Для обработки прерываний по времени предусмотрены организационные блоки OB10-OB17.

Прерывания с задержкой

- Имеет приоритет 3, для обработки этих прерываний предусмотрены организационные блоки ОВ20- ОВ23. Эти блоки вызываются в том случае, если произошло заданное событие и истекло запрограммированное время задержки. Вызов блоков происходит однократно. Например, двигатель отключается по расписанию, а через 20 минут отключается вентилятор, охлаждающий этот двигатель.

Периодические прерывания

- Для обработки этих прерываний предусмотрены организационные блоки OB30 - OB38. При помощи этих блоков реализуется режим вызова управляющей программы (в соответствующей части) через заданные интервалы времени, начиная с первого цикла обработки управляющей программы. От циклического режима отличается тем, что длительность цикла может меняться, но не превышать 100 мс. Для периодических же прерываний строго заданы промежутки времени: от 5 с для блока OB30 (приоритет 7, кратность 500) до 10 мс для блока OB38 (приоритет 15, кратность 1).

Периодические прерывания

- Чем чаще вызывается блок (меньше период вызова блока), тем меньше времени tб должно затрачиваться на его обработку, но всегда tб < 0,5 tп, где tп – время периода вызова.
- Эти прерывания (соответствующие им блоки) используются для обработки функций цифрового регулирования, так как связаны с шагом дискретизации по времени. Для каждого блока должен задаваться фазовый сдвиг - параметр, который позволит исключить одновременную обработку нескольких прерываний в начале какого-либо интервала. Для блока OB30 сдвиг может быть от 0 до 499, а для OB38 – всегда 0.

Аппаратные прерывания

- Используются для обработки ситуаций, когда требуется быстрая реакция контроллера на событие. Могут вырабатываться модулями ввода/вывода дискретных и аналоговых сигналов (сигнальными модулями), если в них предусмотрен соответствующий режим, коммуникационными процессорами, принявшими срочное сообщение, другими функциональными модулями, выполняющими специализированные функции (например, регулирование).
- Для обработки аппаратных прерываний можно запрограммировать один из блоков от OB40 (приоритет 16) до OB47 (приоритет 23). Порядок обработки прерываний следующий:

Аппаратные прерывания

- при возникновении прерывания вызывается соответствующий организационный блок;
- в организационном блоке анализируется причина возникновения прерывания.

- Например, если прерывание вызвано сигнальным модулем PB8, но в блоке обработки прерывания выполняется команда L PB8, производящая считывание его состояния, после чего выявляется бит, который изменился. Обычно с этим битом связан какой-либо функциональный блок, который вызывается, выполняет необходимые действия и выполняет, например, команду T PB20 – вывод управляющего сигнала.

Аппаратные прерывания

- Основное отличие обработки аппаратных прерываний состоит в том, что все входные или выходные сигналы считываются (или выводятся) непосредственно с периферийных модулей, минуя области отображения. Это занимает длительное время, поэтому количество сигналов, обрабатываемых в этом режиме, не должно быть большим, а сами эти прерывания используются, в основном, для обработки аварийных ситуаций.

Однократный запуск

- Это запуск программ, которые обрабатываются однократно при переходе контроллера из состояния STOP в состояние RUN. Различают 3 режима перезапуска:
  - «теплый перезапуск» (возврат питания) – это режим, в котором контроллер переходит в состояние STOP по причине отключения питания. Для обработки этого режима служит функциональный блок OB101.
  - «полный перезапуск» - это режим, в котором контроллер из состояния STOP в состояние RUN при переключении переключателя RUN. Для обработки этого режима служит функциональный блок OB100.

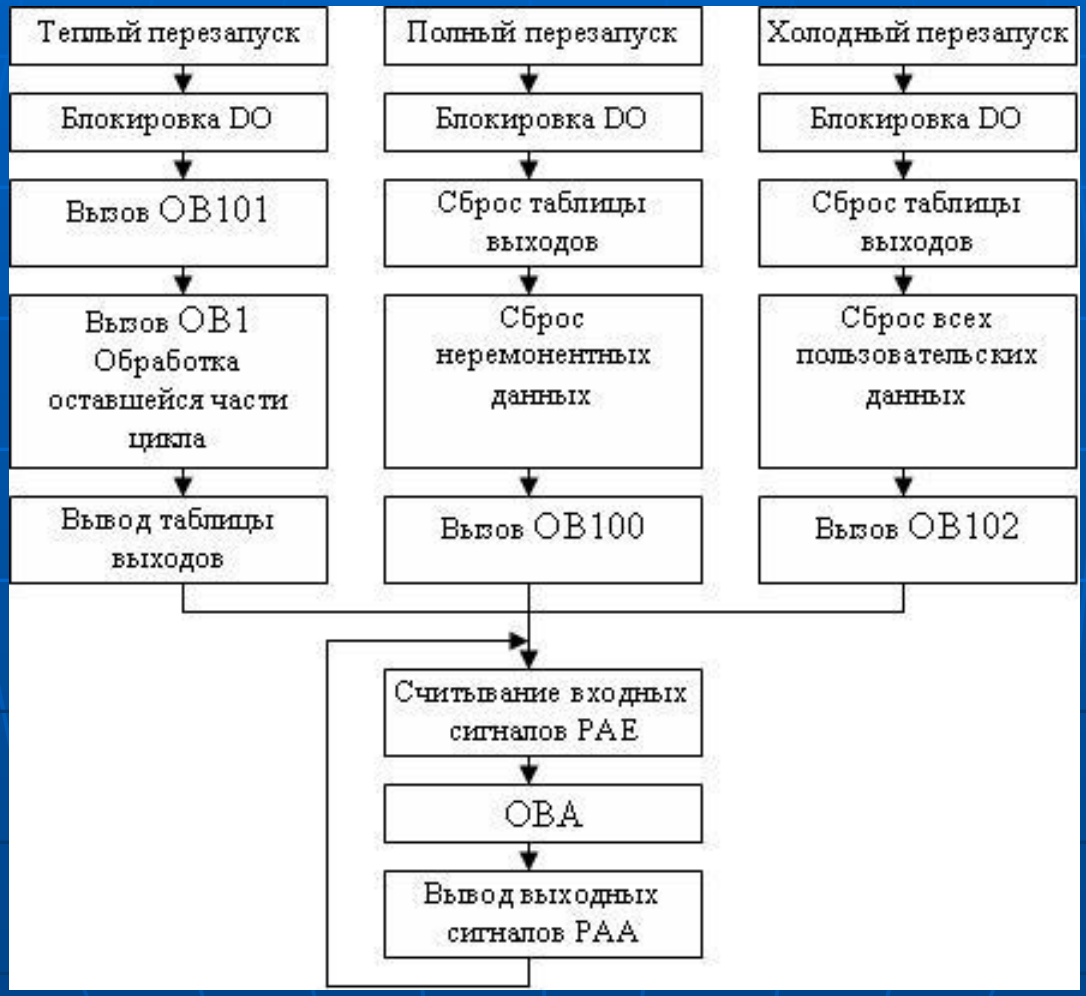
Однократный запуск

- «холодный перезапуск» - это режим, в котором контроллер из состояния STOP в состояние RUN при изменении режимов со стороны устройства программирования. Для обработки этого режима служит функциональный блок OB102.
- Рассмотрим действия, которые выполняет ОС в этих случаях.
- Блоки OB100 и OB102 используются для задания начальных значений переменных, а также синхронизации работы CPU с коммуникационными процессорами и функциональными модулями. Для этих целей предназначен стандартный функциональный блок FB SYNCHRON, который вызывается столько раз, сколько коммуникационных процессоров и функциональных блоков установлено.

Однократный запуск

- Здесь под ремонентными данными (переменными) имеются в виду переменные(счетчики, датчики, меркеры), сохраняющие свое состояние после выключения контроллера, а неремонентными – те, значения которых сбрасываются

Однократный запуск



Обработка ошибок

- В процессе работы контроллера могут возникать следующие ошибки:
- 1) связанные с ошибками при выполнении отдельных операций (деление на ноль, переполнение). Такие ошибки выявляются при помощи анализа флагов результата выполнения операции непосредственно в управляющей программе и организации переходов (условного вызова блоков).
- 2) ошибки выполнения управляющей программы (вызов несуществующего функционального блока, выход за диапазон блока данных).

Обработка ошибок

- Такие ошибки называются синхронными и обрабатываются операционной системой. ОС вызывает блок OB120- OB122, в которых можно запрограммировать реакцию на ошибку или (если блок отсутствует в памяти, то есть не был запрограммирован) переводит контроллер в состояние STOP.
- 3) аппаратные ошибки (возникающие в контроллере и не связанные с управляющей программой). Такие ошибки называются асинхронными и обрабатываются операционной системой, вызывающей блок OB80- OB87, или останавливающей контроллер, если такие блоки отсутствуют. В случае асинхронных прерываний после вызова блока возможны два следствия:

Обработка ошибок

- а) продолжается циклическая обработка управляющей программы;
- б) контроллер переходит в состояние STOP.
- Во втором случае вызванные блоки могут спасти значения переменных управляющей программы.
- Реакции операционной системы контроллера на различные виды ошибок описаны в технической документации.
- Для поиска причин возникновения ошибок в контроллере предусмотрено несколько областей, которые можно проанализировать с помощью программатора, когда контроллер находится в состоянии STOP. Это области:

Обработка ошибок

- BSTEK, в которую заносится информация о последовательности вызова блоков;
- ISTEK, или стек прерываний, позволяет выявить, при обработке какого прерывания возникла ошибка);
- LSTEK, или стек локальных данных;
- Диагностический буфер, в который записывается информация об ошибках.

- Для предупреждения ошибок существует специальный режим отладки управляющей программы по контрольным точкам.

СТРУКТУРА ПРОГРАММ S7

- Программа состоит из логических блоков и блоков данных. Логические блоки бывают: организационными(OB), функциональными(FB) и функциями(FC). **Организационные блоки** выполняют различные задачи.
- Для выполнения основной задачи вам потребуются:
- Блоки запуска (OB100, OB101);
- Блок циклической обработки (OB1) для основной части вашей программы;
- Блоки обработки ошибок (от OB80 до OB87, OB121, OB122), если Вы не хотите, чтобы CPU переключался в STOP при возникновении ошибок;

СТРУКТУРА ПРОГРАММ S7

- организационные блоки для обработки прерываний в CPU или прерываний от процесса;
- Функции и функциональные блоки. OB* можно программировать как структурную программу, создавая функции(FC) и функциональные блоки(FB) и вызывая их в кодовой части OB
- Функциональный блок (FB*) - это логический блок "с памятью". В качестве памяти служит при этом соответствующий функциональному блоку экземпляр блока данных, в котором хранятся фактические параметры и статические данные функционального блока.

СТРУКТУРА ПРОГРАММ S7

- Функция (FC*) - это логический блок "без памяти", иными словами, без соответствующего экземпляра DB. После обработки FC его выходные параметры содержат рассчитанные значения функции. Дальнейшее использование и сохранение фактических параметров после вызова полностью зависит от пользователя.
- Операционная система делает доступными следующие данные
  - Периферийные входы и выходы;
  - Обзор процесса на входах и выходах;
  - меркеры;
  - счетчики;
  - таймеры;

СТРУКТУРА ПРОГРАММ S7

- Можно также определить свои собственные данные:
- глобальные данные (доступны всей программе пользователя);
- статические переменные (действительны только в функциональном блоке, где они определены). При каждом вызове функционального блока указывается экземпляр блока данных, который кроме всех параметров содержит также статические данные. Если определена многоэкземплярная модель, то экземпляры данных, включая статические данные, хранятся в экземпляре блока данных.

СТРУКТУРА ПРОГРАММ S7

- временные данные. Эти данные требуют только стековой памяти во время текущей обработки логического блока.
- Блоки данных хранят данные программы пользователей. Существует два типа блоков данных: глобальные и экземпляры блоков данных. К глобальным блокам данных возможен доступ из всех блоков программы. Экземпляры блоков данных ставятся в соответствие функциональным блокам и содержат помимо данных FB также данные определенных при необходимости мультиэкземпляров.

СТРУКТУРА ПРОГРАММ S7

- Программа пользователя в основном состоит из блоков. Программа пользователя может содержать следующие элементы: OB, FB, FC, DB*.
- Для облегчения работы можно создавать свои собственные типы данных, определенные пользователем (UDT – User-defined Data Type), которые могут использоваться или как типы данных в собственном смысле, или как шаблон для создания блоков данных.
- Некоторые из часто используемых блоков, такие, как системные функциональные блоки (SFB*) и системные функции (SFC*), встроены в CPU.

Программирование с использованием символов

- В таблице символов назначаются символические имена и типы данных всем абсолютным адресам, к которым можно позднее обращаться в вашей программе. Используя символическое представление, вы облегчите себе задачу создания и читаемости программы написанной вами.
- Для заполнения таблицы символов откройте компонент Simbols[Символы] в окне SIMATIC MANAGER.

Программирование с использованием символов

	Symbol	Address	Data Type
1	Cycle Execution	OB 1	OB 1
2			

Программирование с использованием символов

- В начале таблица символов состоит только из предварительно определенного организационного блока OB1. Щелкнув на Cycle Execution [Исполнение цикла] можно заменить его, для примера, словами "Main Program [Главная программа]".
- В строке 2 введите имя переменной и адрес, например "Green Light [Зеленый свет]" и "Q 4.0". Тип данных добавится автоматически. В столбце Comment [Комментарий] можно ввести комментарий к символу.

Программирование с использованием символов

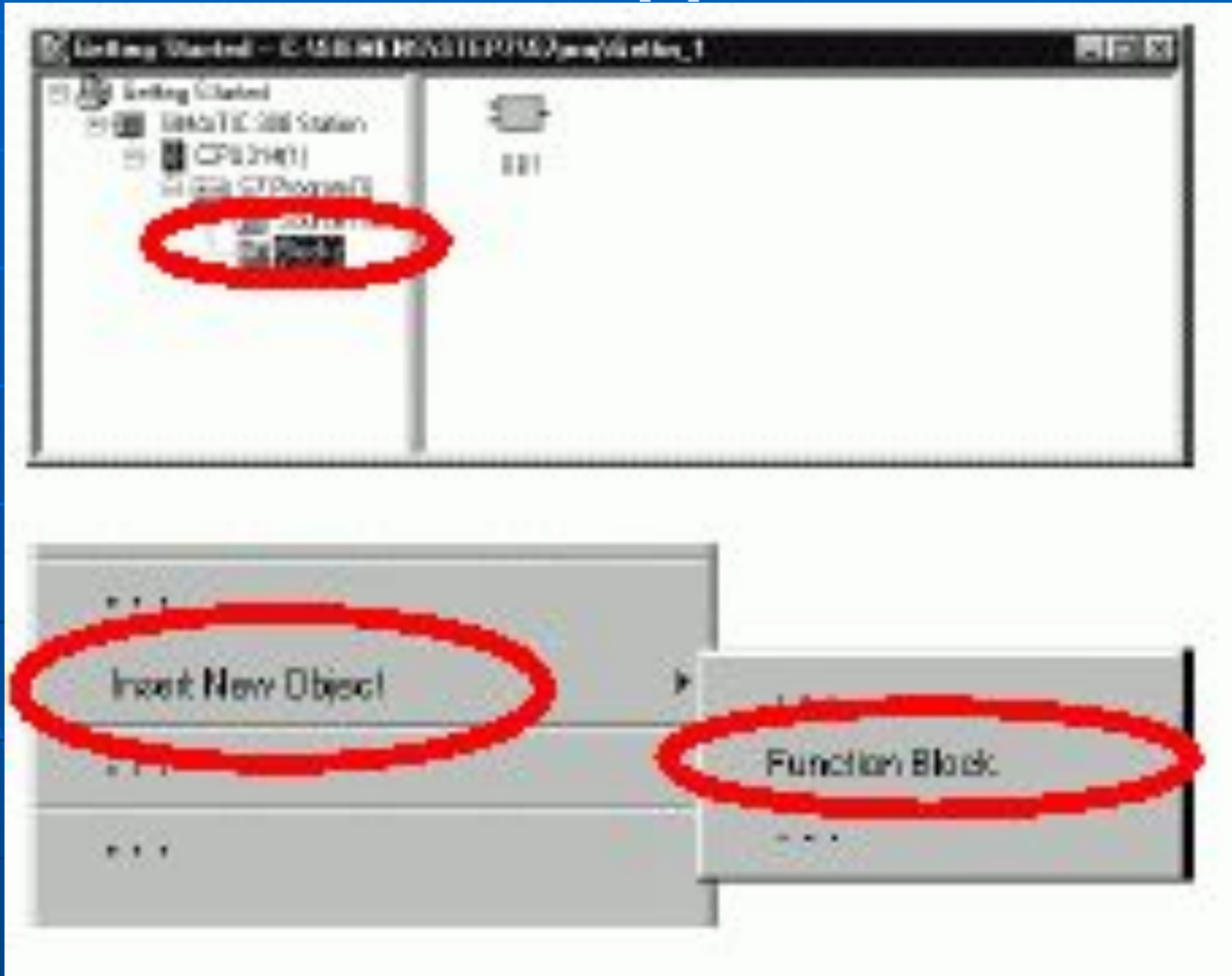
	Symbol	Address	Data Type
1	Main Program	OB 1	OB 1
2	Green Light	Q 4.0	BOOL

Comment

Программы с функциональными блоками и блоками данных

- Функциональный блок (FB*) содержит часть программы, которая может многократно вызываться OB1. Все формальные параметры и статические данные функционального блока сохраняются в отдельном блоке данных (DB*), назначаемом функциональному блоку.
- В папке Blocks [Блоки] SIMATIC MANAGER нужно выбрать в контекстном меню **Insert New Object>Function Blocks** [**Вставить новый обьект> Функциональный Блок**]

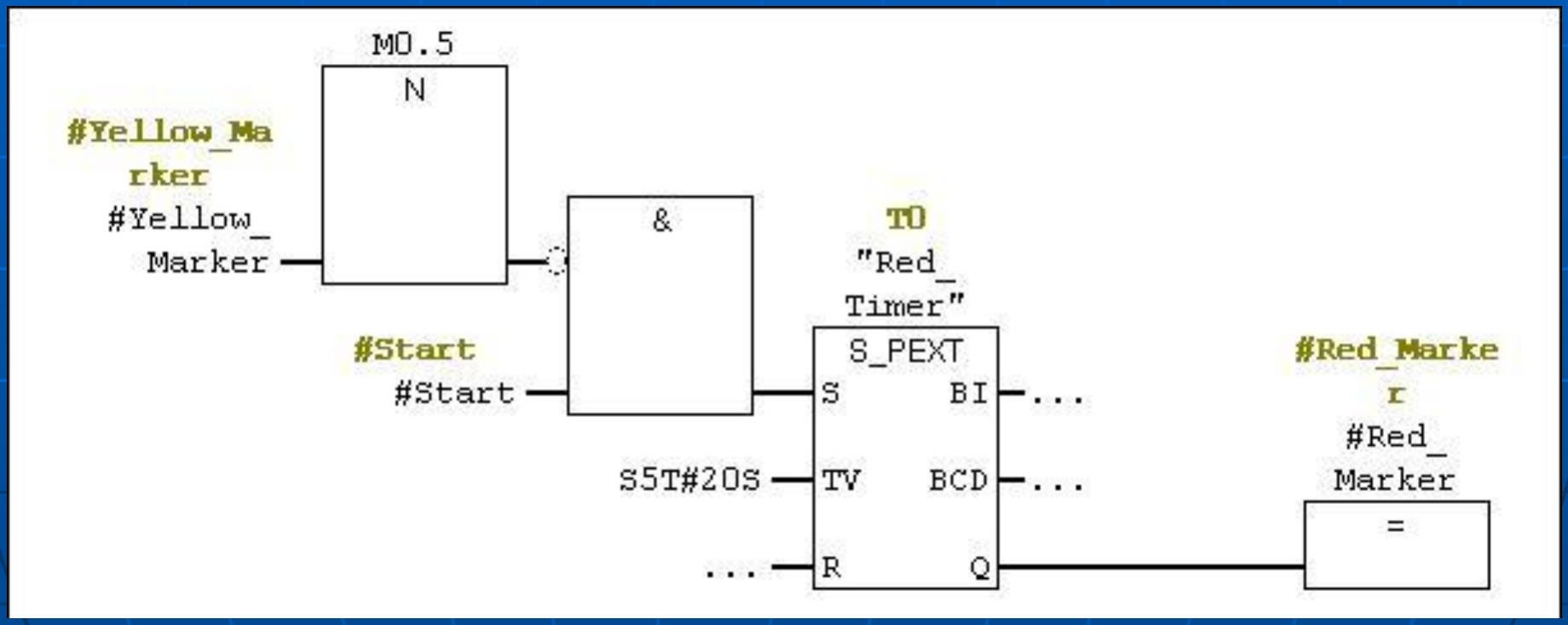
Программы с функциональными блоками и блоками данных



Программы с функциональными блоками и блоками данных

- Далее надо открыть FB1 и в диалоговом окне **"Properties –Function Block [Свойства – Функциональный блок]"** выбрать язык, на котором надо создавать этот блок, **снять** триггерную кнопку "Multiple instance FB [Мультиэкземплярный FB]" и подтвердить остальные параметры настройки, щелкнув на OK.
- В блоке необходимо ввести описание входных (IN) и выходных (OUT) параметров блока.
- Затем, на любом из языков запрограммировать блок.

Программы с функциональными блоками и блоками данных



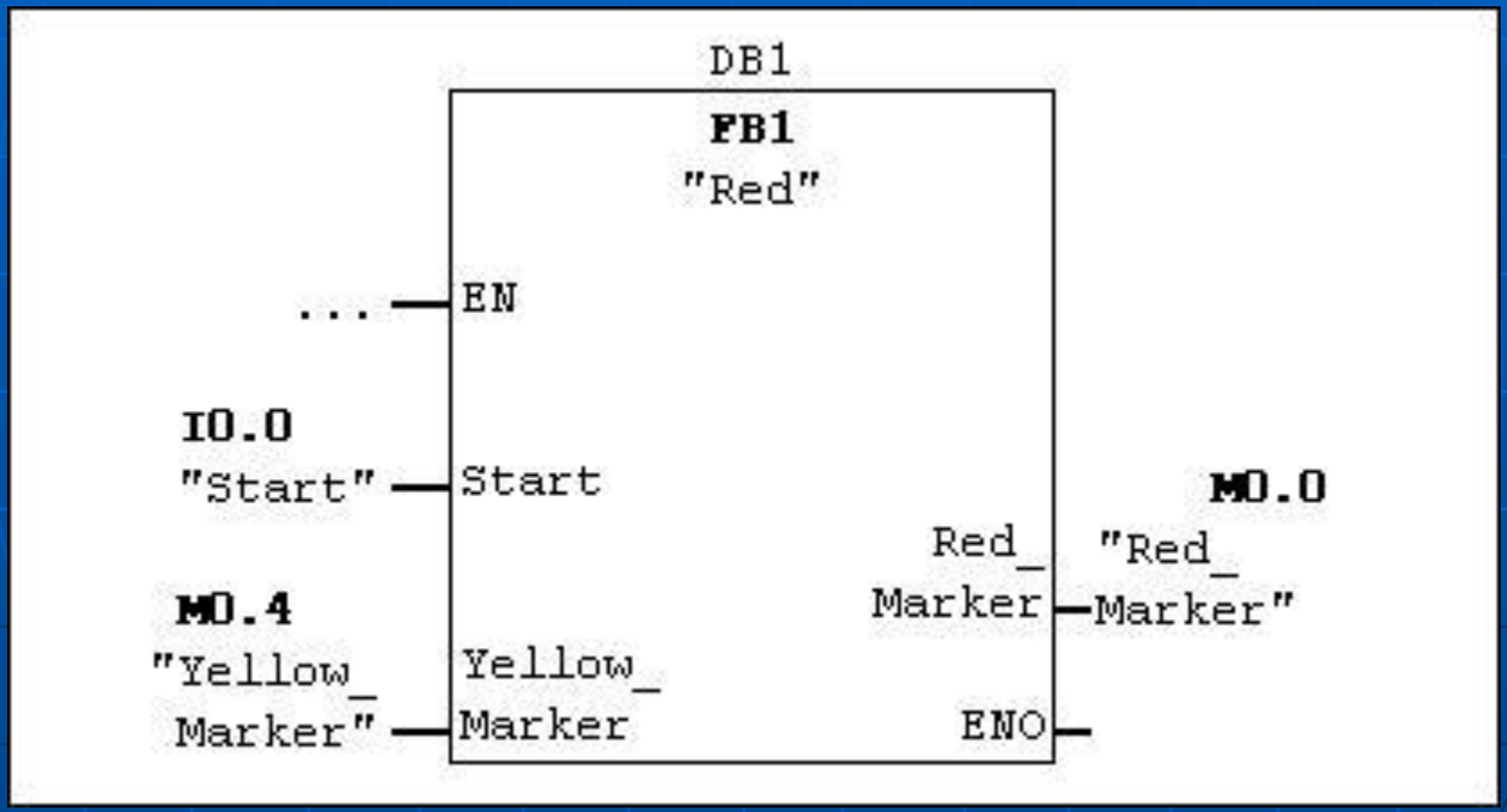
Программы с функциональными блоками и блоками данных

- Чтобы в будущем получить возможность вызова функционального блока в OB1, вы должны сгенерировать соответствующий блок данных. Экземплярный блок данных (DB) всегда ставится в соответствие функциональному блоку.
- Централизованно программируя функциональный блок один раз, вы можете сократить объем программирования.
- Для этого надо создать блок данных DB в папке **Блоки** в SIMATIC MANAGER и принять все параметры, отображаемые в диалоговом окне **Propeties**.

Программирование вызова блока

- Создавая структуры программ с организационными блоками, функциональными блоками и блоками данных, необходимо программировать вызов для подчиненных блоков (таких, как FB1) в блоке, расположенном в иерархии более высоко (например, OB1).
- Возможно также давать различным блокам символические имена в таблице символов (например, FB1 имеет имя "Engine [Двигатель]", а DB1 – имя "Petrol [Бензиновый]").
- При вставке в OB1 функционального блока необходимо указывать соответствующий ему блок данных и передавать/принимать **фактические** параметры блока.

Программирование вызова блока



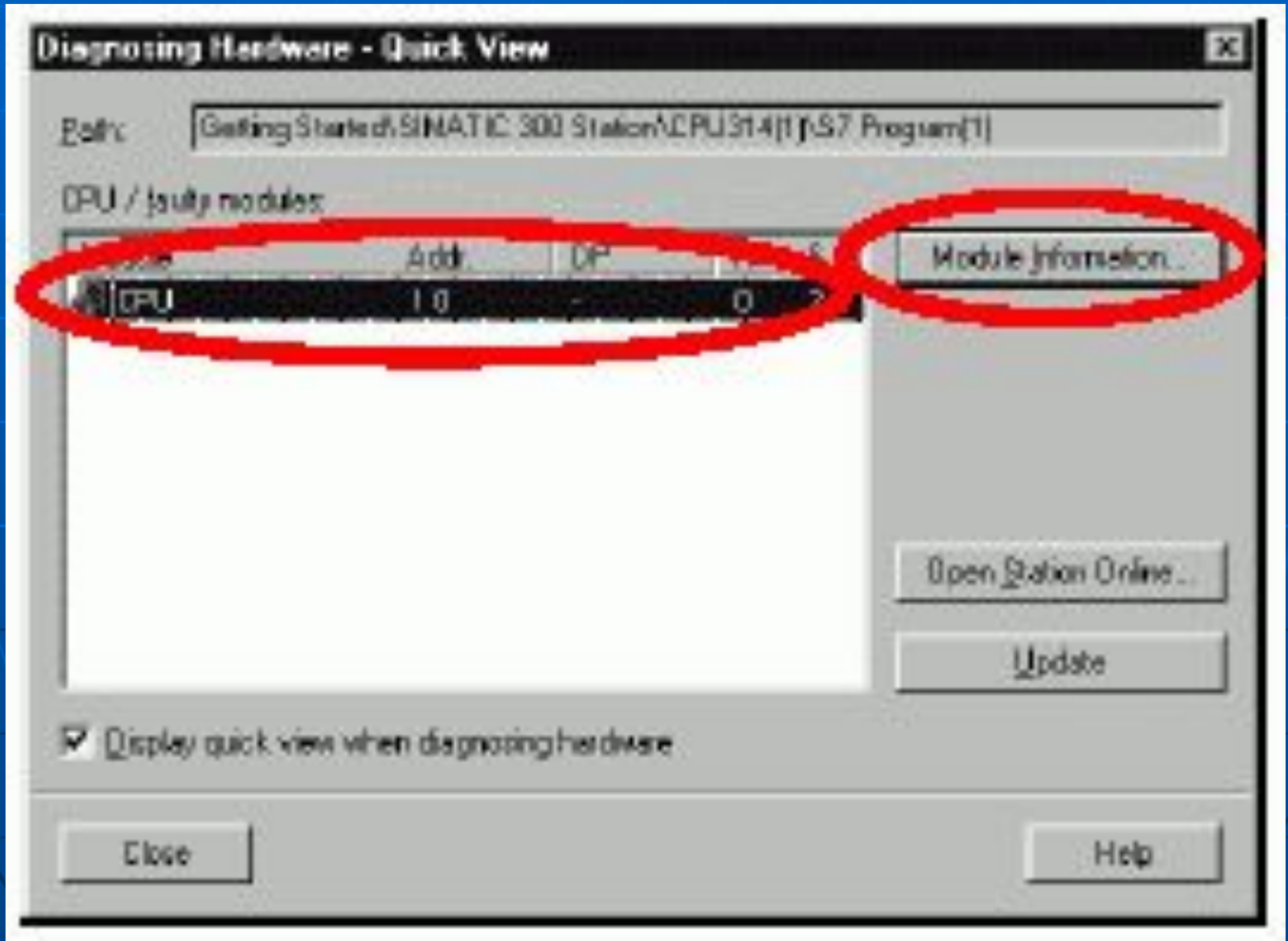
Поиск и анализ ошибок

- Если CPU переходит в STOP при обработке программы S7, или невозможно переключить CPU в RUN после загрузки программы, то можно определить причину ошибки из событий, перечисленных в диагностическом буфере.
- Наиболее частая причина – не загруженный функциональный блок или блок данных.
- Предпосылкой для этого является установление связи online с CPU и нахождение CPU в состоянии STOP.

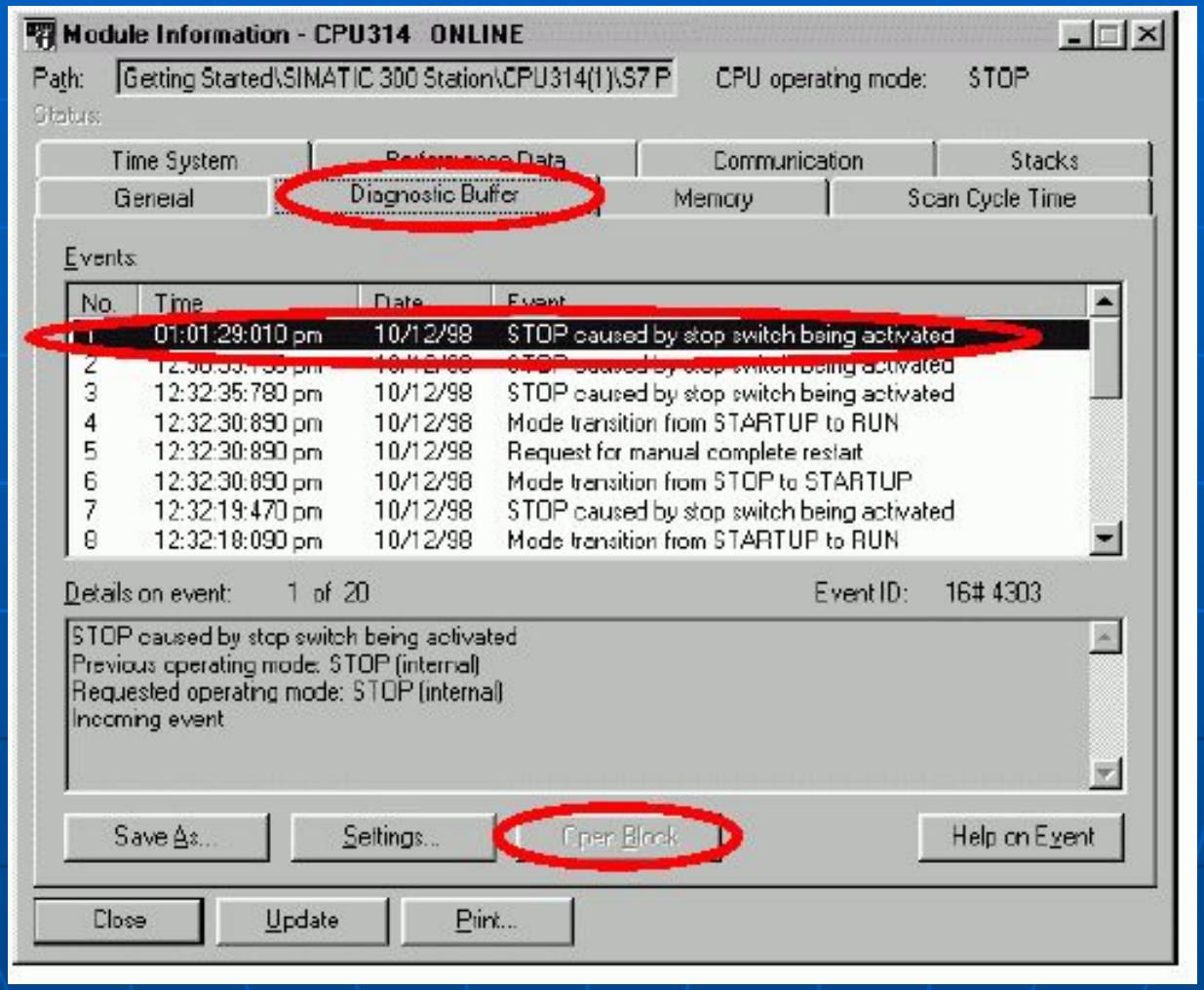
Поиск и анализ ошибок

- Все доступные CPU перечислены в диалоговом окне **"Diagnosing Hardware [диагностирование аппаратуры]"** из меню **PLC**. CPU, находящееся в состоянии STOP, выделены подсветкой. Щелкнем на кнопке **Module Information [Информация о модуле]**, чтобы проанализировать диагностический буфер этого CPU.
- Если подключен только один CPU, то можно запросить информацию о модуле для этого CPU непосредственно с помощью команды меню **PLC > Module Information [ПЛК > Информация о модуле].**
- Окно **"Module Information"** предоставляет информацию о свойствах и параметрах CPU. Теперь можно выбрать вкладку **"Diagnostic Buffer",** чтобы определить причину перехода в состояние STOP.

Поиск и анализ ошибок



Поиск и анализ ошибок



Поиск и анализ ошибок

- Самое последнее событие (номер 1) находится наверху списка. Отображается причина перехода в состояние STOP.
- Если переход CPU в состояние STOP вызвала ошибка программирования, необходимо выбрать событие и щелкнуть на кнопке "Open Block [Открыть блок]". В окне для программирования LAD/STL/FBD открывается блок, и сегмент, содержащий ошибку, выделяется подсветкой.