```
///////////////////////////////////////////////////////////////////////////////
// TcAsyncBufferWritingModule.h
#pragma once

#include "TcIoInterfaces.h"
#include "TcAsyncBufferWritingDrvServices.h"
#include "TcAsyncBufferWritingDrvInterfaces.h"

#include "TcRtInterfaces.h"  //to get systime
#include <string>
using namespace std;

#include "TcFileAccessInterfaces.h"
#include "TcFsmFileWriter.h"

#define ASYNCWRITE_ContiBUFFERSIZE 100
#define ASYNCWRITE_EventBUFFERSIZE 20
///////////////////////////////////////////////////////////////////////////////
// CTcAsyncBufferWritingModule has two buffers for one mode,
// which are filled with current data alternately. The member m_pBufferFill
// points to the current fill buffer. If a buffer is completely filled, then
// the member m_pBufferWrite is set such that it points to the filled buffer.
// This data is written to a file using the TcFsmFileWriter.
class CTcAsyncBufferWritingModule
    : public ITComObject
    , public ITcADI
///<AutoGeneratedContent id="InheritanceList">
    , public ITcCyclic
///</AutoGeneratedContent>
{
public:
    DECLARE_IUNKNOWN()
    DECLARE_IPERSIST(CID_TcAsyncBufferWritingDrvCTcAsyncBufferWritingModule)
    DECLARE_ITCOMOBJECT_LOCKOP()
    DECLARE_OBJDATAAREA_MAP()
    DECLARE_ITCADI()

    CTcAsyncBufferWritingModule();

    virtual ~CTcAsyncBufferWritingModule();

///<AutoGeneratedContent id="InterfaceMembers">
    // ITcCyclic
    virtual HRESULT TCOMAPI CycleUpdate(ITcTask* ipTask, ITcUnknown* ipCaller, ULONG_PTR context);

///</AutoGeneratedContent>
    struct st_Buffer        //structure of Buffer
    {
        double Timestamp;
        double setRoomT1;
        double setRoomT2;
        double setWaterT;
        double setOutT;
        double watertemp;
        double roomtemp1;
        double roomtemp2;
        double a;
        double b;
        double b2;
    };

protected:
    DECLARE_ITCOMOBJECT_SETSTATE();

    HRESULT AddModuleToCaller();
    VOID RemoveModuleFromCaller();
```

```
///<AutoGeneratedContent id="Members">
    TcAsyncBufferWritingModuleParameter m_Parameter;
    TcAsyncBufferWritingModuleInputs m_Inputs;
    TcAsyncBufferWritingModuleOutputs m_Outputs;
    ITcCyclicCallerInfoPtr m_spCyclicCaller;
///</AutoGeneratedContent>

    BOOL FillBuffer
    (
        st_Buffer* pBuffer,
        UINT nBuffer,
        UINT& nBufferIndex
    );

    LONGLONG systime;
    char buf[1024] = "\0";
    char bu[1024] = "\0";
    PCCH name;
    st_Buffer m_Buffer1[ASYNCWRITE_ContiBUFFERSIZE];
    st_Buffer m_Buffer2[ASYNCWRITE_ContiBUFFERSIZE];        //Buffer1 and Buffer2 for continue logging
    st_Buffer m_Buffer3[ASYNCWRITE_EventBUFFERSIZE];
    st_Buffer m_Buffer4[ASYNCWRITE_EventBUFFERSIZE];        //Buffer3 and Buffer4 for eventbased logging

    st_Buffer* m_pBufferFill;       //pointer pointing to Buffer to be filled
    st_Buffer* m_pBufferWrite;      //pointer pointing to Buffer to be read in binary file
    UINT   m_nBufferFillIndex;      //Index of record in buffer in continue logging
    UINT   m_nBufferFillIndex2;     //Index of record in buffer in eventbased logging

    char m_szTaskSysTime[32];
    BOOL titleflag;

    //content of title (value names)
    string A = { "timestamp,setroomtemp1,setroomtemp2,setwatertemp,setoutsidetemp,watertemp,roomtemp1,     ↙
    roomtemp2,a,b,b2" };
    char* Title = &A[0];

    ITcFileAccessAsyncPtr m_spFileAccessAsync;
    CTcFsmFileWriter m_fsmFileWriter;           //CTcFsmFileWriter coming from Beckhoff C++ Sample20, to    ↙
    write data into files
    CTcFsmFileWriter m_fsmFileTitleWriter;

    bool Conti1_Event0;
    bool EventFlanke;
    bool init;
    bool m_Event;
    bool EventLastCycle;
    bool m_EventBufferFillFlag;
    UINT Flag;
};
```