X-Bus Intensivseminar

ExDac PC-DLL

SORCUS Computer GmbH
Dipl.-Ing. Jürgen Schäfer

SORCUS

Inhalt

- Aufgabenstellung
- Interner Aufbau
- Klasse Cmeasurement
- PC-Puffer
- Schnittstelle zum Anwenderprogramm
- Schnittstelle zum Echtzeitprogramm
- LabVIEW Anbindung
- Start/Anzeige der Messdaten mit ExDac.exe

Aufgabenstellung

- Initialisierung der Basiskarte
- OsX laden
- Installieren des OsX MDDs
- Installieren des Echtzeitprogramms
- Konfigurieren des Echtzeitprogramms
- Starten der Messung
- Abholen der Messdaten vom Echtzeitprogramm
- Speichern der Messdaten
- Bereitstellung der Messdaten für andere PC-Programme

Interner Aufbau

- Windows 32 DLL
- Die Funktionalität ist in der C++ Klasse *CMeasurement* implementiert
- Die von der Karte empfangenen Messdaten werden in einem Puffer auf dem PC gespeichtert
- Schnittstelle zum PC-Anwender-Programm
  - InitMeasurement
  - ReadAin
- Schnittstelle zum Echtzeitprogramm
  - MeasurementService (Service-Routine zum Empfang der SRQs des Echtzeitprogramms)

CMeasurement

- Member-Variablen:
  - speichern alle relevanten Daten (Kartennummer, Slot-/Layernummer der Module, Abtastfrequenz, Puffer, ...)
- Funktionen:
  - *Konstruktor*
    - Initialisiert die Member-Variablen
  - *Init*
    - führt komplette Initialisierung und Start der Messung durch (Kartenreset, anlegen der PC-Puffer, ...)
  - *Destruktor*
    - löscht Speicher der in *Init* angefordert wurde
- Beim Start einer neuen Messung wird jeweils ein Objekt dieser Klasse angelegt

CMeasurement::Init()

```
ULONG CMeasurement::Init()
{
    // reset board
    Error = max_reset_board(m_rcBoard.usNumber, m_rcBoard.usTimeout);

    // connect X-MAX-1
    Error = max_connect_cpu( m_rcBoard.usNumber,
                             m_rcXmax1.usSlot,
                             m_rcXmax1.usLayer,
                             &m_rcXmax1.rcOsInfo,
                             &m_rcXmax1.hModule);
    // load ROM OsX
    Error = max_load_osx(m_rcXmax1.hModule, "ROM");

    // load X-AD14-20 MDD
    Error = max_load_mdd(    m_rcXmax1.hModule,
                             m_rcXad1420.usSlot,
                             m_rcXad1420.usLayer,
                             0,
                             0x8024,
                             NULL,
                             &m_rcXad1420.hMdd);
```

SORCUS

CMeasurement::Init()

```
// set user service routine
Error = max_set_service(m_rcXmax1.hModule, MeasurementService);

// install rt measurement task
Error = max_transfer_and_install(m_rcXmax1.hModule, &m_rcRtProgram, &usTask);

// allocate OsX Buffer
rcBufferType.usStrategy = MAX_ALLOC_DOWN_MAX;  // allocate as many as possible
rcBufferType.usAlignment = 2;
rcBufferType.usTask = 0;
rcBufferType.usUsage = 0;
m_rcOsXBuffer.ulSize = 100000*sizeof(OSX_BUFFER_DATA);
Error = max_create_buffer(m_rcXmax1.hModule,
                          &rcBufferType,
                          &m_rcOsXBuffer.ulSize,
                          &m_rcOsXBuffer.hBufHandle);
```

CMeasurement::Init()

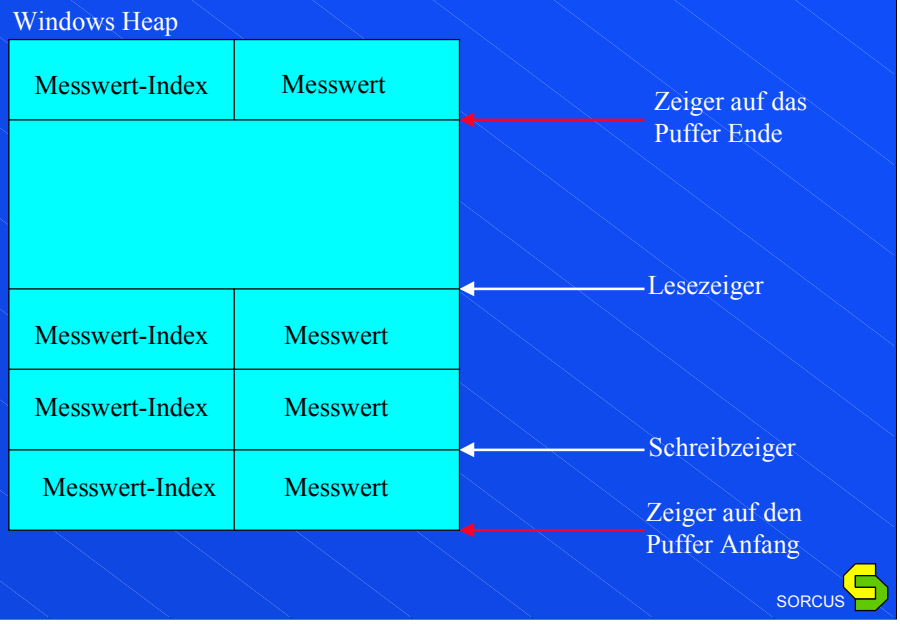
```
    // allocate PC data buffer for AIN-x channel
    for (i=0; i<2; i++)
    {
        m_arcPcAinBuffer[i].prcStart = (PC_AIN_BUFFER_DATA*)
              malloc(m_arcPcAinBuffer[i].ulSize*sizeof(PC_AIN_BUFFER_DATA));
    }

    // set parameter of rt measurement task
    rcInitData.ulAcquisitions = m_ulNumberOfData;
    Error = max_get_buffer_id(m_rcOsXBuffer.hBufHandle, &rcInitData.ulBufferID);
    rcInitData.usADSlot = m_rcXadl420.usSlot;
    rcInitData.usTimeValue = (USHORT)(1189200.0 / m_dFrequency);
    usInsize = sizeof(DAQ_INIT_DATA);
    usOutsize = 0;
    Error = max_call_func(m_rcXmax1.hModule, m_rcRtProgram.usTask, 2, &usInsize,
    (void*)&rcInitData, &usOutsize, NULL);

    // start measurement
    Error = max_call_proc(m_rcXmax1.hModule, m_rcRtProgram.usTask, 3);

}
```

SORCUS

PC-Puffer

- Für jeden Kanal existiert ein eigener Puffer
- Werden in CMeasurement::Init() angelegt
- Werden in CMeasurement::~ CMeasurement() gelöscht
- Sind als Ringpuffer implementiert
- Werden auf dem Windows Heap angelegt
- Zugriffe (lesen/schreiben) sind über Kritische Bereiche gegeneinander verriegelt

PC-Puffer


Windows Heap
Messwert-Index
Messwert
Zeiger auf das Puffer Ende
Lesezeiger
Messwert-Index
Messwert
Messwert-Index
Messwert
Schreibzeiger
Messwert-Index
Messwert
Zeiger auf den Puffer Anfang
SORCUS

Schnittstelle zum Anwenderprogramm

```
extern "C" __declspec(dllexport) unsigned long _stdcall InitMeasurement
(ULONG ulBoardNumber, ULONG ulXmax1Slot, ULONG ulXad1420Slot, ULONG
    ulNumberOfData, double dFrequency)
{
    // init critical section object
    InitializeCriticalSection(&g_CriticalSection);

    // create new measurement object
    g_pMeasure = new CMeasurement(ulBoardNumber,
                                  ulXmax1Slot,
                                  ulXad1420Slot,
                                  ulNumberOfData,
                                  dFrequency);

    // call init function
    ulError = g_pMeasure->Init();
}
```

SORCUS

Schnittstelle zum Anwenderprogramm

```
extern "C" __declspec(dllexport) unsigned long _stdcall ReadAin
(ULONG ulChannelIndex, LONG* plData, ULONG* pulDataIndex)
{
    EnterCriticalSection(&g_CriticalSection);

    // read data from PC-buffer
    ...

    LeaveCriticalSection(&g_CriticalSection);

    return ulReturn;
}
```

SORCUS

Schnittstelle zum Echtzeitprogramm

```
MAX_CALLBACK MeasurementService(MAXMODHND hModul, ULONG ulMessage)
{
    OSX_BUFFER_DATA     arcOsXBufferData[1000];   // buffer for reading OsX buffer

    EnterCriticalSection(&g_CriticalSection);

    // check srq
    if (ulMessage == SRQ_FINISHED)
    {
        bClearOsXBufferComplete = TRUE;
    }
    else if (ulMessage == SRQ_DATA_READY)
    {
         bClearOsXBufferComplete = FALSE;
    }
    else
    {
        // stop measurement
        Error = max_call_proc(g_pMeasure->m_rcXmax1.hModule,
                              g_pMeasure->m_rcRtProgram.usTask,
                              4);
    }
```

SORCUS

Schnittstelle zum Echtzeitprogramm

```
    // get data from OsX buffer
    ulSize = 1000*sizeof(OSX_BUFFER_DATA);
    ulBuffer = ulSize;
    Error = max_get_buffer(g_pMeasure->m_rcOsXBuffer.hBufHandle,
                           MAX_BUFFER_MOVE_MAX,
                           &ulSize,
                           &arcOsXBufferData);

    // write data into PC buffer
    ...

    LeaveCriticalSection(&g_CriticalSection);
}
```

SORCUS

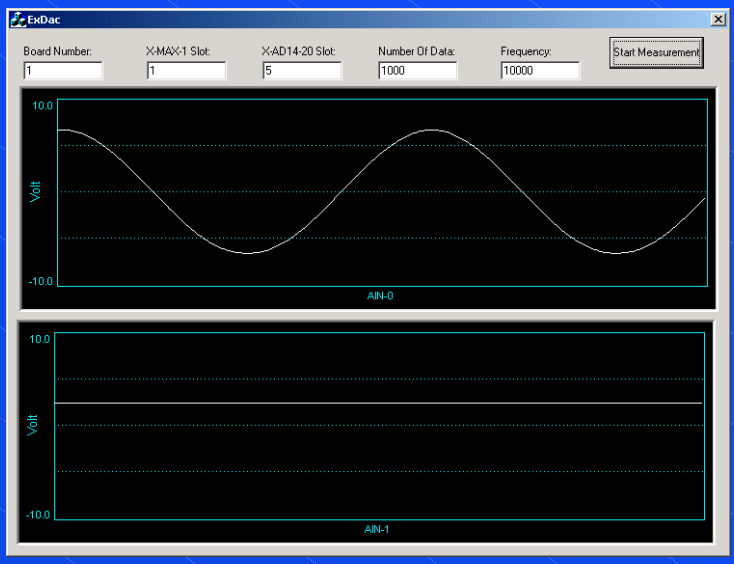
LabVIEW Anbindung

- Zur Anbindung an LabVIEW sind VIs verfügbar, die direkt die entsprechenden DLL Funktionen aufrufen:
  - InitMeasurement
  - ReadAin
- Einfache Beispielanwendung:
  - ExDac

ExDac.exe

- Dialogbasierte PC-Anwendung
- Konfiguriert und Startet die Messung
- Zeigt die erfassten Messdaten an
- Ruft direkt die Funktionen der DLL auf

ExDac.exe



SORCUS

Ende

SORCUS