1 Einführung

Die Achsensteuerungskarten APCI-8001 und APCI-8008 stellen in der Option „SCANNER“ eine Möglichkeit zur Verfügung, um in Echtzeit Prozessdaten aufzuzeichnen. Die maximale Aufzeichnungsrate ist die Abtastzeit der Steuerung (Standardwert: 1,28 ms).
Prozessdaten der Steuerung sind z.B. Sollpositionen, Istpositionen, Ein- oder Ausgangszustände, Achsstati und viele mehr. Alle lesbaren Ressourcen, die im Handbuch „Ressourcen-Interface“, Kapitel 2.2 aufgelistet sind, können aufgezeichnet werden. Die Aufzeichnung erfolgt in einem Speicherbereich der Steuerung. Da dieser Speicher natürlich begrenzt ist, muss bei umfangreicheren Scans der Datenbereich ständig ausgelesen werden, um einen Datenüberlauf zu verhindern. Auf diese Weise ist es jedoch möglich, auch bei umfangreichen Scan-Datensätzen eine Pufferung von mehreren Sekunden zu erreichen. Dadurch ist die Datenaufzeichnung von der Systemauslastung unter Windows weitgehend entkoppelt.
Die Option SCANNER ist nicht in allen Varianten von RWMOS.ELF verfügbar. Die Verfügbarkeit kann im Monitor-Screen von fwsetup.exe verifiziert werden.

2 Konfiguration und Zugriffe auf das Scanner-Modul

Vor der Nutzung des Scanner-Moduls muss dieses konfiguriert werden. Der Zugriff erfolgt mit Hilfe des „Universellen Objekt-Interface“. Diese Zugriffsmethode ist im Handbuch „Universelles Objekt-Interface“ beschrieben.
Die Eigenschaften des Scanner-Moduls und die Vorgehensweise bei der Konfiguration werden im Handbuch „Scanner-Interface“ beschrieben.

3 Verwendung des Scanner-Moduls

Anhand der Programmiersprache C werden nachfolgend die wesentlichen Bestandteile in einem Applikationsprogramm beschrieben, um die Scanner-Funktionalität zu verwenden. Die Standardmethoden zum Initialisieren und Booten der Steuerungsbaugruppe werden als bekannt vorausgesetzt und hier nicht gesondert erwähnt.

3.1 Variablendeklarationen

Zunächst sind die sogenannten Object-Descriptor-Elemente zu deklarieren, mit welchen auf den Scanner bzw. auf die zu scannenden Ressourcen zugegriffen wird. Hierbei handelt es sich um einen vordefinierten Struktur-Datentyp.

Beispiel:

```
// Objekt Descriptors for Ressource Interface
struct OptionDescriptorObject
        // Ressourcen Objekte deklarieren
        odResClean, odRpA1, odDpA1, odScntr, odDigi, odRpA2, odRpA3, odDpA2, odDpA3;
// Objekt Descriptors for Scanner Interface
struct OptionDescriptorObject
        // Scanner-Objekte deklarieren
        odScannerClean, odScannerHW_Scan_Strobe,
        odScanRpA1, odScanDpA1, odScanRpA2, odScanDpA2, odScanRpA3, odScanDpA3,
        odScanScntr, odScanInit, odScannerRTS, odScannerStart,
        odScannerScanned, odScannerSizeOfRec, odScannerStatus, odScanDigi;
```

3.2 Initialisierung der Objekt-Descriptor-Elemente

Die deklarierten Elemente müssen einmalig beim Programmstart anhand der Tabellen in den entsprechenden Handbüchern initialisiert werden. Hierzu müssen folgende Strukturelemente zugewiesen werden:

AccessType
DataType
BusNumber
DeviceNumber
Index
SubIndex

In nachfolgendem Beispiel erfolgt diese Zuweisung in der Funktion SetupOD().

```
/*  Function to initialize the object descriptors
        Parameters: Pointer to od
                            AccessType, DataType, Bus, Device, Index, SubIndex
*/
void SetupOD (struct OptionDescriptorObject *od, ATAccessType at, ATDataType dt,
                int Bus, int Device, int Index, int SubIndex)
{
        od->Handle              = 0;
        od->AccessType          = at;
        od->DataType            = dt;
        od->BusNumber           = Bus;
        od->DeviceNumber        = Device;
        od->Index               = Index;
        od->SubIndex            = SubIndex;
}

// Ressourcen initialisieren
SetupOD (&odResClean,       ATAccessOutput, ATDataDoubleWord, 1000,   0, 1, 0);
SetupOD (&odRpA1,           ATAccessInput,      ATDataReal,           1000,   2, A1, 0);
SetupOD (&odDpA1,           ATAccessInput,      ATDataReal,           1000,   1, A1, 0);
SetupOD (&odRpA2,           ATAccessInput,      ATDataReal,           1000,   2, A2, 0);
SetupOD (&odDpA2,           ATAccessInput,      ATDataReal,           1000,   1, A2, 0);
SetupOD (&odRpA3,           ATAccessInput,      ATDataReal,           1000,   2, A3, 0);
SetupOD (&odDpA3,           ATAccessInput,      ATDataReal,           1000,   1, A3, 0);
SetupOD (&odScntr,          ATAccessInput,      ATDataDoubleWord,     1000,   5, 0, 0);
SetupOD (&odDigi,           ATAccessInput,      ATDataDoubleWord,     1000,   4, 0, 0);

SetupOD (&odScannerClean, ATAccessOutput, ATDataDoubleWord, 1100, 0, 1, 0);
SetupOD (&odScannerRTS, ATAccessOutput, ATDataDoubleWord, 1100, 0, 7, 0);
SetupOD (&odScanDpA3, ATAccessInputOutput, ATDataNone, 1100, 4, 1, odDpA3.Handle);
SetupOD (&odScanRpA3, ATAccessInputOutput, ATDataNone, 1100, 5, 1, odRpA3.Handle);
.....
usw.
```

Dem Element „Handle“ wird zunächst der Wert 0 zugewiesen. Bei der erstmaligen Verwendung wird dieser Wert vom System verwendet, um bei wiederholten Aufrufen die Zuordnung zum entsprechenden Objekt in der RWMOS-Betriebssystemsoftware herstellen zu können.
Bei einem Boot der Steuerung während des Programmablaufs müssen deshalb alle Handle-Elemente abgenullt werden.

3.3 Initialisierung des Scanner-Moduls

Nun können mit Hilfe der initialisierten Object-Descriptor-Elemente die Variablen des Scanner-Moduls beschrieben und gelesen werden. Zum Lesen und Schreiben stehen in mcug3.dll die Funktionen wrOptionInt(), rdOptionInt() für Ganzzahlvariablen und rdOptionDbl und wrOptionDbl für Double-Gleitpunktzahlen zur Verfügung.
Achtung: Hier muss genau darauf geachtet werden, dass die Funktion, welche dem zu lesenden / schreibenden Datentyp entspricht, verwendet wird. Des Weiteren sollte der Rückgabewert immer ausgewertet werden, um den Erfolg der gewünschten Operationen zu gewährleisten und etwaige Fehler an der richtigen Stelle erkennen zu können.
Ansonsten gestaltet sich eine Fehlersuche bei einem nicht funktionierenden Programm als äußerst schwierig.

Beispiel:

```
// Cleaning the ressources
int ival = 1;
if (wrOptionInt (&odResClean, &ival) != 4) return -1;
```

3.4 Initialisierung der zu scannenden Ressourcen

Auf die zu scannenden Ressourcen-Elemente muss vor der Verwendung zumindest ein erfolgreicher Lesezugriff durchgeführt werden, damit das Handle-Datenelement der jeweiligen Ressource vom System initialisiert ist.

Beispiel:

```
// Lesevorgang durchführen, damit man jeweils ein gültiges Handle erhält
if (rdOptionDbl (&odRpA1,  &dval) != 4) return -1;
if (rdOptionDbl (&odDpA1,  &dval) != 4) return -1;
if (rdOptionDbl (&odRpA2,  &dval) != 4) return -1;
if (rdOptionDbl (&odDpA2,  &dval) != 4) return -1;
if (rdOptionDbl (&odRpA3,  &dval) != 4) return -1;
```

3.5 Definition der Scan-Liste

Mit den so vorbereiteten Ressourcen kann man nun die Objekt-Deskriptoren für die Scan-Liste initialisieren und die Scan-Liste, d.h. die Liste der zu scannenden Objekte erstellen.

Beispiel:

```
ival = 0;
SetupOD (&odScanDpA1, ATAccessInputOutput, ATDataNone, 1100, 4, 1, odDpA1.Handle);
ErrorStatus |= wrOptionInt (&odScanDpA1, &ival);

ival = 0;
SetupOD (&odScanRpA1, ATAccessInputOutput, ATDataNone, 1100, 5, 1, odRpA1.Handle);
ErrorStatus = wrOptionInt (&odScanRpA1, &ival);
```

Besondere Hinweise: Der Datentyp für Elemente der Scan-Liste ist „ATAccessInputOutput“.
Die Objekt-Descriptoren für die Scan-Liste können erst initialisiert werden, wenn gültige Handles für die zu scannenden Objekte vorliegen.
In den Scan-Daten erscheinen die entsprechenden Datenobjekte in umgekehrter Reihenfolge, wie sie programmiert werden, d.h., das Element, welches zuletzt in die Scan-Liste eingetragen wird, steht in den Scan-Daten an erster Stelle.

3.6 Starten des Scans

Nachdem die Scan-Liste programmiert ist, können Zustandsinformationen zur Systemüberprüfung ausgelesen werden.
Zumindest sollte man den Scanner-Status (siehe Funktion 4 „STATUS“ bzw. Kommando rdScannerStatus) und die Datensatzgröße des Scan-Records überwachen (siehe Funktion 9 „SIZEOFRECORD“). Vor dem Auslesen dieser Informationen muss die Funktion 2 „INIT“ beschrieben werden, wodurch die entsprechenden Variablen in RWMOS.ELF aufbereitet werden.
Nach fehlerfreier Initialisierung kann der Scan gestartet werden durch Schreiben des Werts 1 auf die Variable 3 „STARTSTOP“.

3.7 Auslesen der gescannten Daten

Wenn ein Endlos-Scan durchgeführt wird, oder wenn nicht gewährleistet ist, dass der Scan-Speicher auf der APCI-8001 bzw. APCI-8008 die gesamten zu scannenden Daten aufnehmen kann, muss nun zyklisch der Scan-Speicher ausgelesen und verarbeitet oder im PC-Arbeitsspeicher oder auf der Festplatte abgelegt werden. Das Auslesen des Scan-Speichers erfolgt mit der DLL-Funktion rdScannerBuffer. Auch hier sind die Rückgabewerte der Funktion unbedingt auszuwerten, um etwaige Fehler sofort zu erkennen.
Die APCI-8001 bzw. APCI-8008 verfügt standardmäßig über einen Scan-Speicherbereich von 100.000 Bytes. Mit der Umgebungsvariablen „SZSCANBUFFER“ kann dieser Wert vorgegeben werden. Maximal möglich sind hier ca. 12 MByte. Der Maximalwert ist aber von anderen Optionen und von der verwendeten RWMOS.ELF-Variante abhängig.
Die Scan-Daten werden auf der APCI-8001 bzw. APCI-8008 in einen Ringpuffer geschrieben. Durch das Auslesen der Daten wird in diesem Ringpuffer immer wieder der entsprechende ausgelesene Datenbereich frei.
Ein etwaiger Datenüberlauf kann mit dem Overrun-Bit im Scanner-Status erkannt werden.
Das Auslesen der Daten erfolgt am besten in einen Speicherbereich, welcher als Array aus Scan-Datensätzen deklariert ist.
Da sich in der Praxis zeigte, dass das Auslesen der Scanner-Daten manchmal sehr viel Zeit benötigt, wird ab mcug3.dll V2.5.3.107 und rwmos.elf V2.5.3.126 die neue Funktion SendReqScannerBuffer() zur Verfügung gestellt. Hierbei handelt es sich um eine Sendeanforderung an rwmos.elf. Bei dieser Art der Datenübertragung kann die Übertragungszeit signifikant reduziert werden. Allerdings ist bei dieser Methode die erfolgreiche Anforderung von physischem Arbeitsspeicher erforderlich. Erfahrungsgemäß wird dieser Speicher nicht zuverlässig vom Windows Betriebssystem bereitgestellt, so dass nach einigen Programmaufrufen bzw. bei starker Auslastung des Windows Betriebssystems die Funktion nicht genutzt werden kann.
Um diese Einschränkungen zu umgehen, kann der Windows Service rwPhysMemService mit Hilfe des Programms rwMemMgnt.exe installiert werden. Dieser Service wird auf der Toolset-CD mitgeliefert und verwaltet den notwendigen Speicher über die gesamte Laufzeit des Windows Betriebssystems.

3.8 Stoppen des Scans

Ein Scan mit einer festen Anzahl von Scan-Datensätzen (Variable 7 „RecordsToScan“) stoppt automatisch, sobald die programmierte Anzahl von Datensätzen aufgenommen wurde. Ansonsten kann ein Scan durch Schreiben von 0 auf die Variable 3 „STARTSTOP“ angehalten werden.
Nach dem Stoppen des Scans sollte darauf geachtet werden, dass alle restlichen Scan-Datensätze der Steuerung ausgelesen werden.