POSITIONIER- UND BAHNSTEUERUNG MCU-G3

G3-Scanner-Interface

Stand: 23.05.2024, ab CD-ROM V2-53VM

1 Einführung

Die Scanner-Funktionalität der MCUG3 Produkte kann verwendet werden um Prozessdaten in Echtzeit zu scannen und zwischenzuspeichern. Hierbei werden die Prozessdaten zyklisch in einem Aufzeichnungsrecord gespeichert. Diese Records können als Recordarrays ausgelesen und verarbeitet werden. Für die Anwendung dieser Funktionalität ist die Verwendung des MCU-G3 Resourcen-Interface erforderlich, welches im File „G3-Resourcen-Interface.pdf“ beschrieben ist.
Diese Option kann nur verwendet werden, wenn eine Betriebssystemvariante RwMos.ELF mit den Optionen *optionSCANNER* und *optionRESOURCE* zur Verfügung steht.

2 Verwendung der Scanner-Funktionen

2.1 Initialisierung des Scanners

Folgende Werte für das Universelle Object-Interface sind für die Verwendung des Scanner-Moduls zu verwenden:

Tabelle 1: Object-Descriptor-Elemente

Object-Descriptor Element	Wert
Handle	Muss beim Start der Applikation oder nach Reboot der Steuerung mit 0 initialisiert sein und wird dann vom System geführt / verwendet. **Bei PCAP-Programmierung:** Nach einem Clean der Scanner-Funktionalität müssen die Handles der Elemente mit rdwr-Funktionalität erneut genullt werden.
BusNumber	1100
DeviceNumber	0
Index	0, 1, ... Funktionsnummer zum Konfigurieren / Bedienen des Scanners lt. Tabelle 2
SubIndex	hier keine Bedeutung

Mit DeviceNumber > 0 und Index 1 werden die Scan-Objekte deklariert. DeviceNumber muss fortlaufend vergeben werden.

Die zu schreibenden Parameter werden beim Aufruf von wrOptionInt (wrOptionDbl) als zweiter Parameter (value) übergeben, bzw. bei der SAP-Programmierung direkt zugewiesen.

Weitere Informationen zu den Object-Descriptor-Elementen sind im Dokument G3-Universelles-Objekt-Interface.pdf beschrieben.

2.2 Funktionen des Scanner-Moduls

Tabelle 2a: Funktionen Scanner-Modul für Device Nr. 0

Nr. (Index)	Bez.	Typ	Erläuterung	Übergabe-parameter (value)
1	CLEAN	integer w	Scanner zurücksetzen Als Wert (value) muss 1 übergeben werden. **Bei PCAP-Programmierung:** Unmittelbar nach dem Rücksetzen müssen die Handles aller verwendeten Objekte mit rdwr-Funktionalität genullt werden.	1
2	INIT	integer w	Scanner vor dem Start initialisieren so wird z.B. SizeOfRecord berechnet und der Datenpuffer wird geleert. **Vorsicht:** Durch diesen Aufruf wird auch die Variable HW_SCAN_STROBE zurückgesetzt.	1
3	STARTSTOP	integer r/w	Scanner starten oder stoppen, bzw. Zustand abfragen	1 = Start 0 = Stop
4	STATUS	integer r	Status des Scanners lesen Der Rückgabewert dieser Funktion wird bei der Funktion rdScannerStatus beschrieben.	
5	SIZEBUFFER	integer r	Größe des gesamten Speichers des Scan-Buffers abfragen (in Bytes). Default: 100.000 Bytes Dieser Wert kann mit der Umgebungsvariable SZSCANBUFFER in fwsetup vorgegeben werden.	
6	TIMEFACTOR	integer r/w	Zeitraster in TA zur Aufzeichnung von Daten lesen / schreiben Defaultwert: 1	1, 2, ...
7	RECORDSTOSCAN	integer w	Anzahl der Datensätze (Records), die aufgezeichnet werden sollen lesen / schreiben. Die Anzahl wird im Übergabewert (value) übergeben. Falls hier 0 eingetragen wird, wird endlos gescannt. Falls mehr Records aufgezeichnet werden sollen, als in den Datenpuffer passen, müssen die aufgezeichneten Daten während des Aufzeichnungsvorgangs ausgelesen werden. Dieser Wert wird beim Scan bis auf 0 abgezählt. Durch erneutes Beschreiben kann der Scan fortgesetzt werden. **Defaultwert: 1**	0
8	RECORDSSCANNED	integer r	Anzahl der aufgezeichneten Records abfragen. Durch Aufruf der Funktion ScannerInit wird dieser Wert auf 0 zurückgesetzt.	

Tabelle 2b: Funktionen Scanner-Modul für Device Nr. 0 (Fortsetzung)

Nr. (Index)	Bez.	Typ	Erläuterung	Übergabe-parameter (value)
9	SIZEOF RECORD	integer r	Größe des aufzuzeichnenden Records abfragen (in Bytes). Dieser Wert steht erst nach Aufruf der Funktion INIT zur Verfügung.	
10	CHECK BUFFER	integer r	Größe des freien Speichers im Scan-Buffer abfragen (in Bytes).	
11	HW_SCAN_ STROBE	integer r/w	Durch Setzen von einem oder mehren Bits in diesem Register, können schnelle Hardware-Eingänge als Strobe-Eingänge für den Latchvorgang definiert werden. Damit diese Option genutzt werden kann, muß RWMOS.ELF mit der entsprechenden Option ausgerüstet sein. **Vorsicht:** Diese Variable wird durch einen Aufruf von INIT zurückgesetzt.	bits 0..7
22	FREE BUFFER	integer w	Interne Funktion für Speicherverwaltung, nicht für den Anwender bestimmt	freizugebender Speicher in Bytes
64	SYNCPULSE OUT	integer r/w	Nur bei spezieller Hardwarevariante: Mit Hilfe dieses Registers kann eine Scannersyncrone Impulsausgabe zur Triggerung externer Komponenten erfolgen (siehe Abschnitt 2.4 Scan-Trigger-Output)	bitcodierte Angabe der Achsen für Impulsausgabe

2.3 Scan-Steuerung

Defaultmäßig wird der Scan zeitgesteuert und abtastsynchron durchgeführt. Es ist aber auch möglich den Scan ereignisgesteuert durchzuführen. Hierzu wird als letztes Scan-Element die Ressource WTLSTRB (#101) für eine definierte Achse definiert. In diesem Fall wird der Scan jeweils dann aufgezeichnet, wenn ein Latch-Strobe-Signal der entsprechenden Achse erkannt wurde. Jedoch auch hier erfolgt der Scan abtastsynchron. Der Parameter TIMEFACTOR sollte hier immer auf 1 stehen. Die Latchimpulse dürfen nicht schneller als die Abtastzeitpunkte anliegen. Das Latch-Strobe-Signal wird beim Aufzeichnen des Scan-Records jeweils zurückgesetzt.

2.4 Scan-Trigger-Output

In einer speziellen Hardwarevariante und mit RWMOS ab V2.5.3.78 ist es möglich, synchron zum Scan ein Hardware Triggersignal auszugeben, welches verwendet werden kann um z.B. externe Komponenten bei der Messwertaufnahme zu synchronisieren. Dieses Signal kann, je nach Hardwarevariante, ein RS-422 oder ein 24V Digitalausgang sein.
Hierzu muß die Scannervariable SYNCPULSEOUT (#64) mit einem bitcodierten Wert beschrieben werden, in welchem die Achsen für die Impulsausgabe durch gesetzte Bits gekennzeichnet sind.
Beispiel: 3. Achse = Impulsausgang, dann muß der Wert 4 in die Variable SYNCPULSEOUT geschrieben werden.
I.A. wird nur ein Ausgang für einen derartigen Zweck vorbereitet sein. Bei der entsprechenden Achse, steht kein Nullspursignal und kein Hardwarelatch zur Verfügung. Bei Verwendung eines 24V Digitalausgangs kann dieser auch auf herkömmliche Weise geschaltet werden. Der tatsächlich ausgegebene Pegel ergibt sich durch Veroderung der angegebenen Zustandsinformationen.
Mit der entsprechenden Hardwarevorbereitung ist es zusätzlich möglich, eine schnelle Impulsausgabe aus der Softwareumgebung über die Ressource #64 (siehe Handbuch „G3-Resourcen-Interface.pdf“) vorzunehmen, d.h. durch Schreiben auf die Ressource #64 kann ein RS422 Ausgang oder, bei entsprechender Hardware-Vorbereitung, ein Digitalausgang sofort betätigt werden.
Hinweis: Die Ausgabe auf Digitale Ausgänge wird nur einmal pro Abtastintervall der Steuerung (normalerweise 1,28ms) aktualisiert. Mit einer schnellen Impulsausgabe kann eine Ausgabe mehrfach im Abtastintervall erfolgen.

2.5 Definition der Scan-Records

2.5.1 PCAP-Programmierung

Die aufzuzeichnenden Daten werden durch Funktionsaufrufe mit nachfolgenden Daten definiert. Die aufzuzeichnenden Elemente müssen zuvor über das G3-Resourcen-Interface definiert werden. Der Aufzeichnungsrecord wird in der umgekehrten Reihenfolge aufgebaut, wie die einzelnen Elemente definiert werden. Alle Elemente des G3-Resourcen-Interface können gescannt werden.

Tabelle 3: Definition des Aufzeichnungsrecords.

Object-Descriptor Element	Wert
BusNumber	1100
AccessType	Input/Output
DataType	(unbedeutend)
DeviceNumber	1, 2, ...
Index	0, 1, ... TIMEFACTOR für dieses Element
SubIndex	Gültiges Handle eines Object-Descriptors des G3-Resourcen-Interface

Hinweise: Die DeviceNumber muss ungleich 0 sein und eindeutig vergeben werden.
Die Initialisierung der Objekte die aufgezeichnet werden sollen erfolgt mit der Zugriffsmethode ATAccessInputOutput (= 3). Der Inhalt des Parameters DataType bei den Object-Descriptoren der aufzuzeichnenden Daten ist unbedeutend. Die Variable TIMEFACTOR, die in Index eingetragen wird, ermöglicht die Datenaufzeichnung nur zu ganzzahligen Vielfachen der Aufzeichnungsintervalle zu scannen. Mit dem Wert 1 wird der Messwert in jedem Aufzeichnungsintervall aufgezeichnet. Mit dem Wert 0, wird der Messwert nie aufgezeichnet. Standardmäßig muss hier der Wert 1 eingetragen werden.

2.5.2 SAP-Programmierung

Für jedes aufzuzeichnende Datum wird ein AT-Spezifizierer deklariert:

Beispiel:

```
var     ScanListItem_rp:        double  AT %MRScannerBus.1.1.0;
var     ScanListItem_axst:      integer AT %MDScannerBus.2.1.0;
var     ScanListItem_digi:      integer AT %MDScannerBus.3.1.0;
```

Dieser Variable muss dann einmalig (pro Start des SAP-Programms) die zu scannende Ressource mit Hilfe des ptr-Operators zugewiesen werden. Diese Zuweisung muss in umgekehrter Reihenfolge angegeben werden, als die Daten im Aufzeichnungsrecord abgelegt werden.

Beispiel:

```
ScanListItem_rp         := ptr(G3R_rp_A1_r);            // real position of axis 1
ScanListItem_digi       := ptr(G3R_digi_A1_r);          // digital inputs
ScanListItem_ain_CH0 := ptr(G3R_ain_CH0_r);                     // analog value channel 0
```

Hierbei ist darauf zu achten, dass die verwendeten Datentypen übereinstimmen.

2.6 Verwendung der Scanner-Funktionen

- Definition der benötigten ObjectDescriptor Elemente
- aufzuzeichnende Ressourcen definieren,
  mindestens eine Leseoperation ausführen, damit ein gültiges Handle vorliegt.
- Scanner CLEAN aufrufen
- Liste der Scan-Objekte definieren
- Anzahl der aufzuzeichnenden Datensätze programmieren per Variable
  RECORDSTOSCAN
- Scanner INIT aufrufen
- Nun kann der Scan gestartet und auch wieder gestoppt werden mit
  Scanner STARTSTOP
- Das Auslesen der aufgezeichneten Daten erfolgt mit dem PCAP-Kommando rdScannerBuffer (siehe unten)
  Hierbei kann jederzeit der Zustand des Scanners abgefragt werden z.B. mit rdScannerStatus.

Bei PCAP-Programmierung: Falls die Funktion ResourceClean wiederholt aufgerufen wird, müssen alle Handles der Resource-Object-Descriptor-Elemente genullt werden.
Falls die Funktion ScannerClean wiederholt aufgerufen wird, müssen alle Handles der Scanner-Object-Descriptor-Elemente mit rdwr-Funktionalität genullt werden.

2.7 PCAP-Funktionen für Scanner-Zugriffe

2.7.1 rdScannerBuffer, read scanner buffer

BESCHREIBUNG:	Diese Funktion kopiert den aktuellen Scannerbuffer der MCU-G3 in einen Speicherbereich der aufrufenden Anwendung und gibt den entsprechenden Speicher auf der Steuerung wieder für den Scan frei.
BORLAND DELPHI:	function rdScannerBuffer (buffer: PChar; size: integer): integer;
C:	int rdScannerBuffer (char *buffer, int size);
VISUAL BASIC:	Function rdScannerBuffer (buffer As String, ByVal size As Long)
PARAMETER:	Der Parameter *buffer* ist ein Zeiger auf einen Speicherbereich der Anwendung der mindestens *size* Bytes groß sein muss. Der Parameter *size* spezifiziert dabei die zu lesende Anzahl von Bytes.
RÜCKGABEWERT:	Anzahl von Bytes die erfolgreich in den Speicherbereich *buffer* kopiert wurden. 0 – wenn keine Daten im Scanner vorliegen -1 – ScannerBuffer zu groß definiert -2 – Systemfehler im Scanner-Modul
ANMERKUNG:	Die Maximalanzahl auszulesender Bytes wird in dieser Funktion automatisch berechnet. Es werden also maximal *size* Bytes oder weniger ausgelesen. Für die nachfolgende Datenauswertung ist es sinnvoll, immer ein Vielfaches der vorgegebenen Recordlänge auszulesen. Der Aufbau der Datenstruktur, in welche die Daten geschrieben werden, muß mit der Selektion der Scan-Objekte übereinstimmen. Die zurückgelieferten Daten sind nicht nach Wortgrenzen ausgerichtet. Zur Ausführung dieses Befehls ist eine spezielle *rwmos.elf* - Software erforderlich. Die Datenübertragungsraten bei diesem Kommand können, je nach Hardware unter 2 MByte / Sekunde liegen. Mit dem neueren Kommando SendReqScannerBuffer() sind deutlich höhere Übertragungsraten erforderlich. Das Kommando ist in Kapitel xxx beschrieben.

2.7.2 rdScannerBufferSize, read scanner buffer size

BESCHREIBUNG:	Diese Funktion liefert die aktuelle Größe des Scannerbuffers auf der MCU-G3 zurück. Der Rückgabewert hat die Einheit Byte. Standardmäßig ist die Buffergröße auf 100000 Bytes voreingestellt. Die Buffergröße kann mit einer Flash-Umgebungsvariable auf bis zu maximal 13 MByte hochgesetzt werden.
BORLAND DELPHI:	function rdScannerBufferSize: integer;
C:	int rdScannerBufferSize(void);
VISUAL BASIC:	Function rdScannerBufferSize() As Long
RÜCKGABEWERT:	Buffergröße des Scannerbuffers in Bytes.
ANMERKUNG:	Zur Ausführung dieses Befehls ist eine spezielle *rwmos.elf* - Software erforderlich.

2.7.3 rdScannerLsm, read scanner left spool memory

BESCHREIBUNG:	Diese Funktion liefert den momentan frei verfügbaren Arbeitsspeicher des Scannerbuffers zurück. Beim Eintrag im Scanner läuft dieser Wert gegen 0. Beim Auslesen des Scanners läuft der Wert gegen ScannerBufferSize.
BORLAND DELPHI:	function rdScannerLsm: integer;
C:	int rdScannerLsm(void);
VISUAL BASIC:	Function rdScannerLsm() As Long
RÜCKGABEWERT:	Frei verfügbarer Arbeitsspeicher des Scannerbuffers in Bytes.
ANMERKUNG:	Zur Ausführung dieses Befehls ist eine spezielle *rwmos.elf* - Software erforderlich.

2.7.4 SendReqScannerBuffer, Send Request scanner buffer

BESCHREIBUNG:	Diese Funktion kopiert den aktuellen Scannerbuffer der MCU-G3 in einen Speicherbereich der aufrufenden Anwendung und gibt den entsprechenden Speicher auf der Steuerung wieder für den Scan frei.
BORLAND DELPHI:	function SendReqScannerBuffer (buffer: PChar; size: integer): integer;
C:	int SendReqScannerBuffer (char *buffer, int size);
VISUAL BASIC:	Function SendReqScannerBuffer (buffer As String, ByVal size As Long)
PARAMETER:	Der Parameter *buffer* ist ein Zeiger auf einen Speicherbereich der Anwendung der mindestens *size* Bytes groß sein muss. Der Parameter *size* spezifiziert dabei die zu lesende Anzahl von Bytes.
RÜCKGABEWERT:	Anzahl von Bytes die erfolgreich in den Speicherbereich *buffer* kopiert wurden. 0 – wenn keine Daten im Scanner vorliegen und wenn kein Fehler vorliegt Negative Werte zeigen Fehler oder Statusinfos an. Eine Beschreibung der Bitcodierung erfolgt nach dieser Tabelle in Kapitel 2.7.4.1.
ANMERKUNG:	Gegenüber der Funktion *rdScannerBuffer* werden die Daten jedoch nocht von der PCI Baugruppe gelesen, sondern von der Steuerung in den Arbeitsspeicher des PC geschrieben. Deshalb ist mit diesem Kommando die Datenübertragungsrate wesentlich höher. Die Maximalanzahl auszulesender Bytes wird in dieser Funktion ebenfalls automatisch berechnet. Es werden also maximal *size* Bytes oder weniger ausgelesen. Für die nachfolgende Datenauswertung ist es sinnvoll, immer ein Vielfaches der vorgegebenen Recordlänge auszulesen. Der Aufbau der Datenstruktur, in welche die Daten geschrieben werden, muß mit der Selektion der Scan-Objekte übereinstimmen. Die zurückgelieferten Daten sind nicht nach Wortgrenzen ausgerichtet. Zur Ausführung dieses Befehls ist eine spezielle *rwmos.elf* - Software erforderlich. Dieses Kommando ist erst verfügbar ab *rwmos.elf* V2.5.3.126 und mit *mcug3.dll* V2.5.3.107.

2.7.4.1 Erläuterung der Rückgabeinformationen bei negativem Wert

Wenn der Rückgabewert negatives Vorzeichen besitzt (Bit 31 gesetzt), dann hat dieser Wert nicht die Bedeutung von übertragenen Bytes, sondern zeigt bitcodierte Status- und ggf. Fehlerinformationen an. Wenn eines der unteren 16 Bits (0..15) gesetzt ist, handelt es sich um eine Fehlerinformation und eine Datenübertragung mit diesem Kommando ist nicht möglich. Die Bits 16 – 30 zeigen nur Statusinfos an, die Verwendung des Kommandos ist trotzdem möglich. Dennoch kann eine explizite Auswertung dieser Bits sinnvoll sein, um die Zuverlässigkeit des Systems beurteilen zu können. Die Fehler und Statusinfo kann sicher ausgelesen werden, wenn *SendReqScannerBuffer* mit *size* = 0 aufgerufen wird. Wenn *size* > 0

angegeben wird und es liegen nur Statusinformationen, aber auch Daten im Scanner vor, dann wird eine Anzahl von Bytes übertragen und im Rückgabewert angezeigt. In diesem Fall ist die Statusinformation nicht verfügbar.

Tabelle: Bitcodierung des Rückgabewertes des DLL Befehls *SendReqScannerBuffer*

Bit	Wert [hex]	Beschreibung
0	**0000 0001**	Der Zugriff auf das Ressourceninterface ist fehlgeschlagen. In diesem Fall werden im Monitor Screen von *fwsetup.exe* zusätzliche Infos ausgegeben.
1-7	**0000 00FE**	derzeit nicht belegt
8	**0000 0100**	Der Versuch Physischen Speicher zu erhalten schlug fehl! *KS_allocPhysMem* lieferte den Fehler KSERROR_NOT_ENOUGH_MEMORY
9	**0000 0200**	Der Versuch Physischen Speicher zu erhalten schlug fehl! *KS_allocPhysMem* lieferte den Fehler KSERROR_CANNOT_ALLOC_PHYSMEM
10	**0000 0400**	Der Versuch Physischen Speicher zu erhalten schlug fehl! *KS_allocPhysMem* lieferte den Fehler KSERROR_CANNOT_MAP_PHYSMEM
11	**0000 0800**	derzeit nicht belegt
12	**0000 1000**	Der Versuch Physischen Speicher zu erhalten schlug fehl! *KS_allocPhysMem* lieferte einen unbekannten Fehler
13-16	**0000 E000**	derzeit nicht belegt
16	**0001 0000**	Der Zugriff auf den Dienst rwPhysMemService ist nicht möglich (Fehler ERR_OpenFileMapping). In diesem Fall wird konventionell probiert physischen Speicher vom Betriebssystem zu erhalten.
17	**0002 0000**	Der Zugriff auf den Dienst rwPhysMemService ist nicht möglich (Fehler ERR_ MapViewOfFile). In diesem Fall wird konventionell probiert physischen Speicher vom Betriebssystem zu erhalten.
18	**0004 0000**	derzeit nicht belegt
19	**0008 0000**	Die hinterlegten Datenstrukturen in mcug3.dll und rwPhysMemService stimmen nicht überein (Versionsproblem). Ein Zugriff ist nicht möglich. In diesem Fall wird konventionell probiert physischen Speicher vom Betriebssystem zu erhalten. In diesem Fall kann es sein, dass die "Bittigkeit" von Service und Anwenderprogramm nicht übereinstimmt!
20	**0010 0000**	Speicherbereichsgröße des Physischen Speichers ist falsch. (Versionsproblem) In diesem Fall wird konventionell probiert physischen Speicher vom Betriebssystem zu erhalten.
21	**0020 0000**	Kein Physischer Speicher im Service vorhanden (Installationsproblem) In diesem Fall wird konventionell probiert physischen Speicher vom Betriebssystem zu erhalten.
22	**0040 0000**	Die Daten-Kennung in mcug3.dll und rwPhysMemService stimmen nicht überein (Versionsproblem). Ein Zugriff ist nicht möglich. In diesem Fall wird konventionell probiert physischen Speicher vom Betriebssystem zu erhalten.
23	**0080 0000**	Der Speicher des Service *rwPhysMemService* ist bereits in Verwendung (markiert). In diesem Fall wird konventionell probiert physischen Speicher vom Betriebssystem zu erhalten.
24-28	**1F00 0000**	derzeit nicht belegt
29	**2000 0000**	allocPhysMem mit KSF_ALTERNATIVE schlug fehl (nur zur Info). Dieses Bit hat keine nennenswerte Bedeutung für den Anwender.
30	**4000 0000**	Das Anwenderprogramm hat keine Administrator Rechte (wird benötigt zum Markieren der Speicher-Belegung). Wenn der Speicher frei ist, wird er dennoch verwendet. Bei Mehrfachaufrufen ist jedoch eine Doppeltbelegung und damit Datenverlust möglich.
31	**8000 0000**	Vorzeichenbit: Dieses Bit zeigt an, dass die Bedeutung dieser Tabelle wirksam ist, wenn dieses Bit nicht gesetzt ist, dann zeigt der Zahlenwert die Anzahl der übertragenen Bytes an

2.7.4.2 Beschreibung und Hinweise zur Handhabung des Befehls

Vor der ersten Abholung von Daten, sollte das Kommando mit size = 0 aufgerufen werden. Hierbei wird probiert physischen Speicher für die interne Verarbeitung des Kommandos zu erhalten. Dieser Speicher steht dann für die gesamte Laufzeit des Programmes zur Verfügung. Problematisch ist es, wenn kein Speicher zur Verfügung steht. Dann kann dieses Kommando nicht verwendet werden. Deshalb sollte unbedingt die Auswertung des Rückgabewertes bei diesem ersten Aufruf durchgeführt werden. Wenn der verwendete Speicher vom Betriebssystem Dienst *rwPhysMemService* bereitgestellt wird, dann wird dieser, wenn dem aufrufenden Programm Administratorrechte zur Verfügung stehen, als "in Verwendung" markiert. Somit ist eine doppelte Verwendung des gleichen Speicherbereiches verhindert. Beim Beenden der Applikation wird diese Markierung automatisch wieder aufgehoben und der Speicher wird für den nächsten Programmaufruf wieder verfügbar gemacht. Wenn der Dienst *rwPhysMemService* nicht erreichbar ist, oder wenn der Speicher dieses Dienstes als "in Verwendung" markiert ist, dann wird probiert Speicher vom Betriebssystem mit dem internen DLL-Kommando *KS_allocPhysMem* zu erhalten.
Wenn die Möglichkeit besteht, dass das Programme gleichzeitig ausgeführt werden, welche das Kommando *SendReqScannerBuffer* verwenden, sollte darauf geachtet werden, dass die Programme mit Administratorberechtigung ausgeführt werden (bei Windows Vista / 7 / 8 / 10 und zukünftige Versionen). Wenn der erste Aufruf ohne Rückgabe eines Fehlerbits erfolgte, dann kann das Kommando in dieser Session uneingeschränkt in gleicher Weise wie *rdScannerBuffer* verwendet werden.

2.7.5 rdScannerStatus, read scanner status

<table>
<tr><td>BESCHREIBUNG:</td><td>Diese Funktion liefert den aktuellen Status des Scanners auf der MCU-G3 zurück.</td></tr>
<tr><td>BORLAND DELPHI:</td><td>function rdScannerStatus: integer;</td></tr>
<tr><td>C:</td><td>int rdScannerStatus(void);</td></tr>
<tr><td>VISUAL BASIC:</td><td>Function rdScannerStatus() As Long</td></tr>
<tr><td>RÜCKGABEWERT:</td><td>Scannerstatus bitkodiert.

Tabelle: Bitkodierter Aufbau des Scanner Status-Wortes
<table>
<tr><th>Bit-Nr.</th><th>Name</th><th>Funktion</th></tr>
<tr><td>0</td><td>empty</td><td>Status-Flag: Scanner ist komplett leer.</td></tr>
<tr><td>1</td><td>full</td><td>Status-Flag: Scanner ist voll. Die vorgegebene Anzahl von Records wurde eingetragen.</td></tr>
<tr><td>2</td><td>inprocess</td><td>Status-Flag: Der Scanner ist zur Zeit in Arbeit.</td></tr>
<tr><td>3</td><td>endless</td><td>Status-Flag: Die Betriebsart Endlosscan ist angewählt.</td></tr>
<tr><td>8</td><td>norecords</td><td>Error-Flag: Es wurden keine Records definiert. Fehler bei Scanlistengenerierung.</td></tr>
<tr><td>9</td><td>overrun</td><td>Error-Flag: Scanner-Überlauf. Der Scanner-Buffer ist voll. Bei Endlosscan wird jew. der zuletzt eingetragene Datensatz verworfen. Wenn kein Endlosscan, wird der Scan gestoppt. und kann keine neuen Records mehr aufnehmen.</td></tr>
<tr><td>10</td><td>config error</td><td>Error-Flag: Es wurde ein Fehler bei der Konfiguration erkannt
- unbekannter Datentyp</td></tr>
<tr><td>11</td><td>Scan Ressource Not Valid</td><td>Error-Flag: Eine Ressource in der Scan-Liste ist ungültig. Ggf. wurde beim Ressourcen-Interface ein Clean durchgeführt.</td></tr>
</table></td></tr>
<tr><td>ANMERKUNG:</td><td>Zur Ausführung dieses Befehls ist eine spezielle rwmos.elf - Software erforderlich.</td></tr>
</table>

3 Der Windows Service rwPhysMemService

Zur uneingeschränkten Nutzung der Funktion SendReqScannerBuffer(), welche zum beschleunigten Übertragen der Scannerdaten von der Steuerung ins PC-Anwenderprogramm dient, sollte der Windows Service rwPhysMemService installiert werden. Dieser Service fordert beim Booten des Windows Betriebssystems physischen Speicher an und verwaltet diesen über die gesamte Laufzeit des Betriebssystems. Somit ist gewährleistet, dass die schnelle Datenübertragung jederzeit genutzt werden kann, auch nach mehreren Programmaufrufen oder bei starker Auslastung des Windows Betriebssystems. Allerdings kann dieser Speicher nicht gleichzeitig von verschiedenen Anwendungen genutzt werden.
Dieser Service ist für 32bit und für 64bit verfügbar. Welche Variante installiert werden muss hängt aber nicht von der Bittigkeit des Betriebssystems, sondern vom Anwenderprogramm ab, welches den Service verwenden will.

3.1 Installation des Windows Service rwPhysMemService

Zur Installation ist die Datei rwMemMgnt.exe in ein lokales Verzeichnis zu kopieren. In diesem Verzeichnis muss das Programm mit dem Parameter /install aufgerufen werden.

rwMemMgnt /install

Dies kann z.B. in einer Eingabeaufforderungsbox erfolgen. Hierbei ist zu beachten, dass je nach Windows Version Administratorrechte zur Verfügung stehen müssen.
Durch diesen Aufruf wird der Dienst installiert aber noch nicht gestartet. Der Start des Dienstes erfolgt durch einen Reboot des Windows Betriebssystems oder manuell in services.msc

3.2 Deinstallation des Windows Service rwPhysMemService

Zur Deinstallation muss die Datei rwMemMgnt.exe (siehe oben) mit dem Parameter /uninstall aufgerufen werden.

rwMemMgnt /uninstall

Dies kann z.B. in einer Eingabeaufforderungsbox erfolgen. Hierbei ist zu beachten, dass je nach Windows Version Administratorrechte zur Verfügung stehen müssen.
Durch diesen Aufruf wird dem Windows Betriebssystem eine Deinstallationsanforderung mitgeteilt. Nach einem Reboot des Betriebssystems wird dann der Dienst deinstalliert und nicht mehr gestartet. Der Stop des Dienstes kann auch manuell in services.msc erfolgen. Ohne Deinstallation wird der Dienst aber nach einem Reboot des Windows Betriebssystems erneut gestartet.