WinAC AccessDB V2

Anwenderdokumentation

V 2.02 • Mai 2013

Applikationen & Tools

Answers for industry.

SIEMENS

Industry Automation und Drives Technologies Service & Support Portal

Dieser Beitrag stammt aus dem Internet Serviceportal der Siemens AG, Industry Automation und Drives Technologies. Durch den folgenden Link gelangen Sie direkt zur Downloadseite dieses Dokuments.

http://support.automation.siemens.com/WW/view/de/59061340

Bei Fragen zu diesem Beitrag wenden Sie sich bitte über folgende E-Mail-Adresse an uns:

online-support.automation@siemens.com

applications.aud.koe.nrh.rd@siemens.com

SIEMENS

SIMATIC
WinAC AccessDB V2

Anwenderdokumentation

Gewährleistung und Haftung

Hinweis

Die Applikationsbeispiele sind unverbindlich und erheben keinen Anspruch auf Vollständigkeit hinsichtlich Konfiguration und Ausstattung sowie jeglicher Eventualitäten. Die Applikationsbeispiele stellen keine kundenspezifischen Lösungen dar, sondern sollen lediglich Hilfestellung bieten bei typischen Aufgabenstellungen. Sie sind für den sachgemäßen Betrieb der beschriebenen Produkte selbst verantwortlich. Diese Applikationsbeispiele entheben Sie nicht der Verpflichtung zu sicherem Umgang bei Anwendung, Installation, Betrieb und Wartung. Durch Nutzung dieser Applikationsbeispiele erkennen Sie an, dass wir über die beschriebene Haftungsregelung hinaus nicht für etwaige Schäden haftbar gemacht werden können. Wir behalten uns das Recht vor, Änderungen an diesen Applikationsbeispielen jederzeit ohne Ankündigung durchzuführen. Bei Abweichungen zwischen den Vorschlägen in diesem Applikationsbeispiel und anderen Siemens Publikationen, wie z.B. Katalogen, hat der Inhalt der anderen Dokumentation Vorrang.

Für die in diesem Dokument enthaltenen Informationen übernehmen wir keine Gewähr.

Unsere Haftung, gleich aus welchem Rechtsgrund, für durch die Verwendung der in diesem Applikationsbeispiel beschriebenen Beispiele, Hinweise, Programme, Projektierungs- und Leistungsdaten usw. verursachte Schäden ist ausgeschlossen, soweit nicht z.B. nach dem Produkthaftungsgesetz in Fällen des Vorsatzes, der groben Fahrlässigkeit, wegen der Verletzung des Lebens, des Körpers oder der Gesundheit, wegen einer Übernahme der Garantie für die Beschaffenheit einer Sache, wegen des arglistigen Verschweigens eines Mangels oder wegen Verletzung wesentlicher Vertragspflichten zwingend gehaftet wird. Der Schadensersatz wegen Verletzung wesentlicher Vertragspflichten ist jedoch auf den vertragstypischen, vorhersehbaren Schaden begrenzt, soweit nicht Vorsatz oder grobe Fahrlässigkeit vorliegt oder wegen der Verletzung des Lebens, des Körpers oder der Gesundheit zwingend gehaftet wird. Eine Änderung der Beweislast zu Ihrem Nachteil ist hiermit nicht verbunden.

Inhaltsverzeichnis

1 Automatisierungsaufgabe

1.1 Übersicht

Einführung/Einleitung

Der existierende Treiber **WinAC AccessDB** ermöglicht es externen Hochsprachen-Programmen direkt auf Variablen der WinAC zuzugreifen (Merker, Datenbausteine, Eingänge, Ausgänge).

Aufgrund der Implementierung des Treibers konnten Applikationen nicht einfach von Windows XP nach Windows 7 migriert werden.

Deshalb wurde der Treiber komplett neu entwickelt – **WinAC AccessDB V2**. Die Schnittstelle für den Anwender ist dabei gleich geblieben. Vorhandene Anwendungen, die WinAC AccessDB V1.x verwenden, sollten ohne Code-Änderung auch mit WinAC AccessDB V2.x laufen.

Es gibt mehrere Gründe, die Applikation komplett neu zu entwickeln:

- Kein direkter Speicherzugriff mehr (z.B. Problem beim Laden von DBs auf verändert Adressen)
- Reduzierung der beteiligten Komponenten
- Keine Verwendung von Kerneltreiber mehr (Windows 7 Sicherheitskonzept)
- Verwenden von Funktionen des WinAC ODK V4.2 (ODK_ReadData / ODK_WriteData)
- Keine Verwendung von Mapping auf physikalischem Speicher mehr (Ressourcen für begrenzt - Langzeitstabilität!)
- Reduzierung FB / OB Aufrufe im WinAC Programm

Die Vorteile von WinAC AccessDB gegenüber alternativen Möglichkeiten des Variablenzugriffs auf die WinAC (z.B. OPC, SAPI-S7) sollen auch in der neuen Version erhalten bleiben:

- Performanter Zugriff auf WinAC Daten
- Sehr einfache Integration in Anwender-Programm
- Keine Verbindungsprojektierung in Step7-Projekt nötig

1.2 Erforderliche Kenntnisse

Für das Verständnis der vorliegenden Dokumentation werden Kenntnisse vorausgesetzt:

- SIMATIC WinAC RTX 2010
- SIMATIC Manager STEP7 V5.5

  oder
- SIMATIC STEP7 V11 SP2 / V12 (TIA Portal)

1.3 Verwendete Hard- und Software-Komponenten

Die Applikation wurde mit den nachfolgenden Komponenten erstellt:

Hardware-Komponenten

- Simatic Microbox IPC 427D
  mit Windows Embedded Standard 7

Standard Software-Komponenten

- SIMATIC WinAC RTX 2010
- SIMATIC Manager STEP 7 V5.5

  oder
- SIMATIC STEP7 V11 SP2 / V12 (TIA Portal)

Beispieldateien und Projekte

Die folgende Liste enthält alle Dateien und Projekte, die in diesem Treiber-Paket verwendet werden.

Tabelle 1-1 Enthaltende Dateien

Komponente	Hinweis
Documentation \ WinAC-AccessDB_Doku_v2xx_DE.pdf Documentation \ WinAC-AccessDB_Doc_v2xx_EN.pdf	Die Dokumentation in Deutsch and Englisch
Driver \ WinAcAccDbSrv.rtdll	Der WinAC Treiber
Driver \ setup.bat	Setup Batch für den Treiber
S7ClassicExample \ AccessDBv2.zip	Step7 Beispielprojekt einschl. des Funktionsbausteins
TiaExample \ AccessDBv2.zip	TIA Portal Beispielprojekt einschl. des Funktionsbausteins
Examples \	Verzeichnis mit Beispielapplikationen in verschiedenen Programmiersprachen

2 Übersicht der Applikation

2.1 Funktionsumfang

Die WinAC Treiber **WinAC AccessDB V2** ermöglicht den einfachen und performanten Zugriff auf beliebige Daten der laufenden WinAC RTX von einer PC-Applikation heraus. Sowohl Applikation als auch die WinAC müssen auf dem gleichen Rechner laufen.

Dieser Treiber kann vom Anwender in seinem Projekt mit den gängigen Entwicklungsumgebungen und Programmiersprachen verwendet werden (Visual Basic 6, Microsoft Visual C++ 6, Delphi, etc.).

Die Schnittstelle der neuen Treiberversion ist kompatibel zur Version V1.x.

Hinweis Nur **eine** Client Applikation kann gleichzeitig auf die Funktion von WinAC AccessDB V2 zugreifen!

Da nicht mehr direkt auf den physikalischen Speicher der WinAC zugegriffen wird, sondern ODK-Funktionen dafür benutzt werden, wird das neue Konzept ein wenig an Performance zur Version V1.x einbüßen. Die Geschwindigkeit sollte vergleichbar sein mit der Version V1.x bei aktiviertem „Safe Mode“.

Dieser Nachteil kann aber auf Grund der gewonnenen Sicherheit und Stabilität in Kauf genommen werden.

2.2 Versionsstand des Treibers

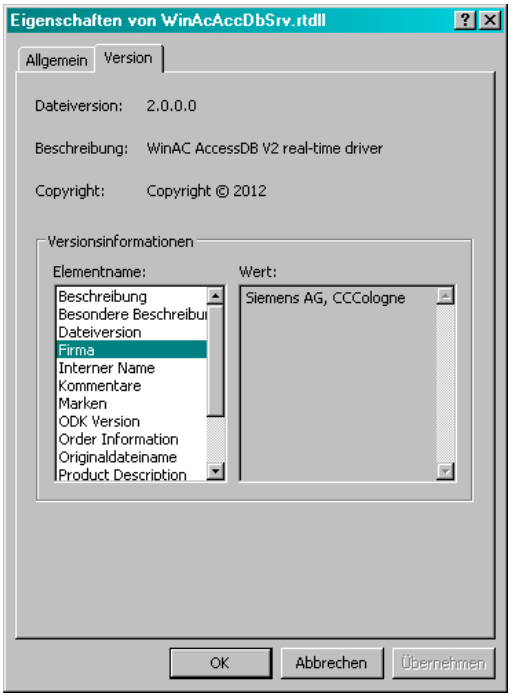
Treiber-Version unter Windows ermitteln

Die registrierte Real-Time Treiber-(RTDLL liegt im Verzeichnis

C:\Windows\RTSS\rtdll**WinAcAccDbSrv.rtdll**

Die Version der Treiber RTDLL kann man im Windows-Explorer über die Dateieigenschaften erfahren (rechte Maustaste → Eigenschaften).

Abbildung 2-1 Versionsstand der Treiber-RTDLL (Windows XP)



Treiber-Version im Step7 Projekt ermitteln

Im Instanz-DB von FB_INIT_ACCESS_DB kann man die Version der Treiber-RTDLL erfahren:

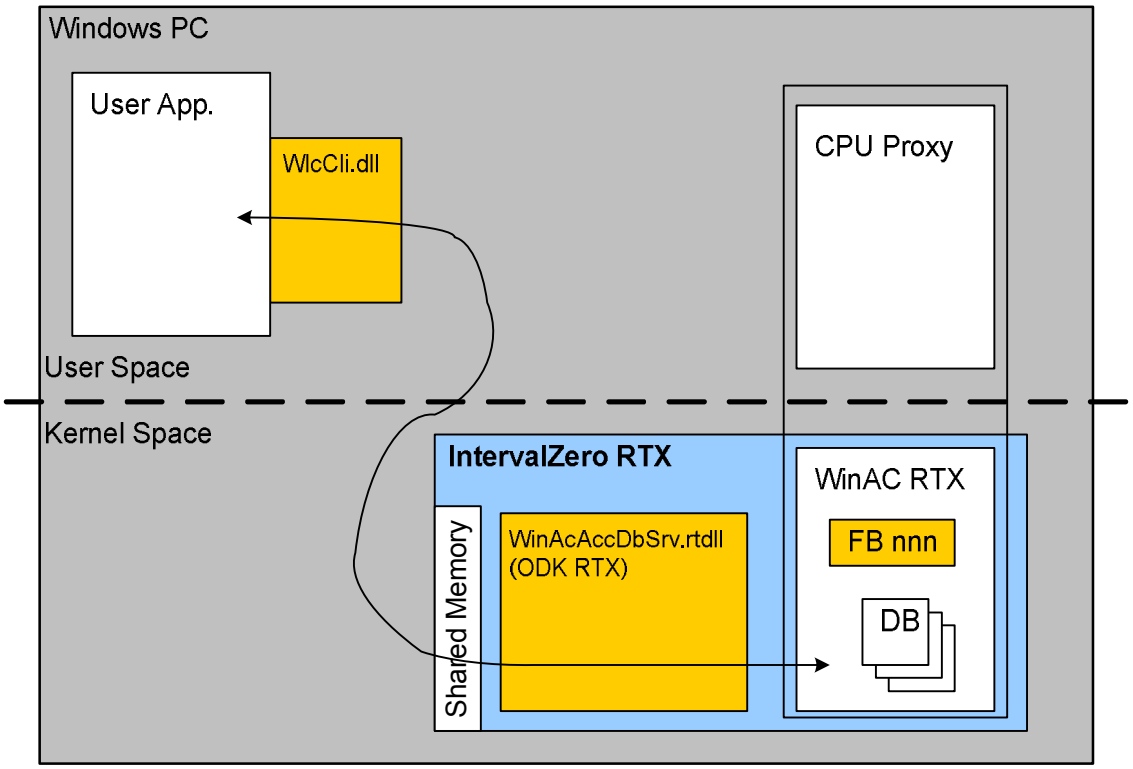
tOdkIf.dwDllVersion Version der Treiber- DLL

3 Funktionelle Beschreibung

3.1 Gesamtübersicht

Bei der neuen Version von **WinAC AccessDB V2** reduziert sich die Anzahl der Komponenten erheblich.

Abbildung 3-1 Strukturbild WinAC AccessDB V2



Die Anwenderapplikation bindet die DLL **WlcCli.dll** ein. Diese DLL kommuniziert über Inter Process Communication (IPC) mit dem ODK-Echtzeit-Treiber **WinAcAccDbSrv.rtdll**.

Im WinAC Programm wird nur ein einziger FB benötigt, der im Anlauf einmalig aufgerufen werden muss. Mit diesem FB wird die ODK-DLL geladen. Zur Laufzeit sind keine FB- oder OB-Aufrufe mehr nötig.

Beispiel - Wie läuft der Zugriff „Lese DB10.DBB 5“ ab?

- WlcCli.dll fordert von WinAcAccDbSrv.rtdll die Daten an:
  Zugriff: „Lesen“
  Type: „Byte“
  Adresse: „DB10, Byte 5“
- WinAcAccDbSrv.rtdll benutzt die ODK-Funktion „ODK_ReadData“ aus dem WinAC ODK V4.2, um die Daten zu beschaffen und gibt diese zurück

Da auf ODK-Funktionen zurückgegriffen wird, ist der Zugriff etwas langsamer, als der direkte Speicherzugriff. Er ist aber wesentlich sicherer, da das Neu-Laden von

DBs nicht zu Fehlern führen kann. Die Performance ist vergleichbar mit der alten Version V1.x bei aktiviertem „Safe Mode“.

3.2 Applikations-DLL (Client)

Die DLL **WinCli.dll** muss in die Hochsprachen-Applikation eingebunden werden.

Aus Anwendersicht stellt diese DLL eine Reihe von Funktionen zur Verfügung wie z.B.: ReadBOOL, WriteBOOL, ReadBYTE, WriteBYTE, etc.

Vor der Verwendung der Lese/Schreib-Funktionen muss die Funktion InitAccessDB() aufgerufen werden. Diese initialisiert den Treiber einschließlich der Interprocess-Communication.

Wird eine Lese-/Schreib-Funktion aufgerufen, wird über IPC (Shared Memory, Events, etc.) eine Anfrage an den ODK-Treiber gestellt. Eine Antwort bzw. ein Timeout Fehler wird an den Aufrufer zurückgeliefert.

Die Funktion SetSafeMode() wird nicht mehr benötigt. Sie ist aber aus Kompatibiliätsgründen vorhanden (ohne Funktion).

3.3 Step7 Funktionen

Der Step7-Treiber besteht nur aus einem einzigen Funktionsbaustein (FB FB_INIT_ACCESS_DB). Der FB muss einmalig (z.B. im OB100 COMPLETE RESTART) aufgerufen werden. Er lädt die Treiber-DLL. Durch das Laden der RTDLL wird der Thread gestartet, der die Funktionalität bereitstellt. Weitere Aufrufe von FBs oder OBs zur Laufzeit sind nicht erforderlich.

Das stellt eine erhebliche Vereinfachung gegenüber der Version V1.x dar.

3.4 ODK-Treiber (Server)

Hinweis Die folgenden Ausführungen zeigen den internen Aufbau des WinAC Treibers. Die RTDLL wird durch die WinAC geladen, d.h. der Hochsprachen-programmierer, der AccessDB V2 verwendet, muss sich mit dieser RTDLL nicht auseinandersetzen.

Der ODK-Treiber ist als Real-Time-Applikation (IntervalZero RTX) realisiert.

Der ODK-Treiber besteht nur aus einem Thread („ODK-Monitor-Thread“), in dem auf Anfragen von der Client-Applikation gewartet wird. Der Auftrag wird entgegen-genommen, bearbeitet und das Ergebnis bzw. Fehlercode zurückgeliefert.

ACHTUNG **Der Server bearbeitet immer nur einen Auftrag vom Client gleichzeitig. D.h. es gibt keine Queue oder Muli-Client-Unterstützung.**

Die Beschränkung auf einen einzelnen Auftrag wurde bewusst gewählt. Auf diese Weise ist die Schnittstelle einfach zu realisieren, weniger fehleranfällig und performanter.

Der WinAC Treiber besteht nur noch aus einer einzigen RTDLL (WinAcAccDbSrv.rtdll). Das stellt eine erhebliche Vereinfachung gegenüber der Version V1.x dar.

3.5 IPC zwischen Anwender- und ODK-Teil

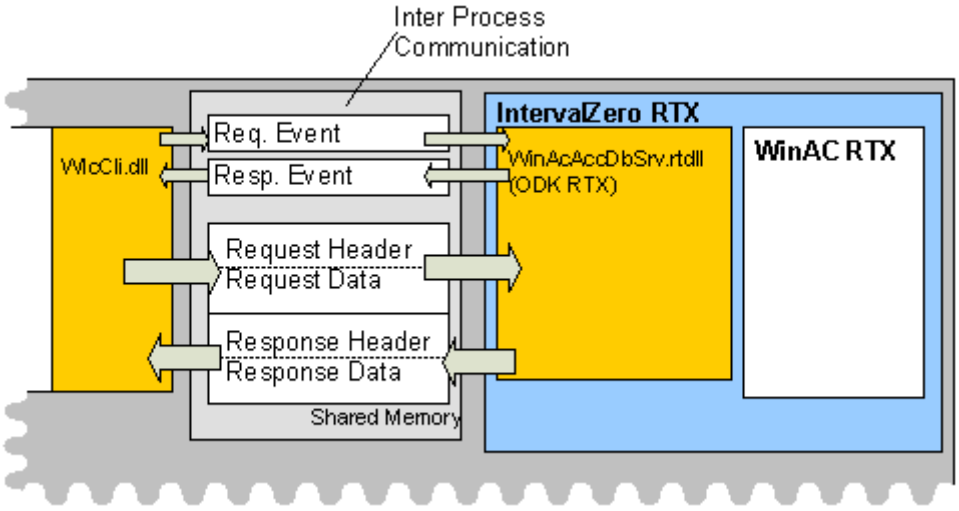
Hinweis Die folgenden Ausführungen zeigen den internen Aufbau des Treibers. Sie sollen dem Verständnis dienen. Für den Hochsprachenprogrammierer, der AccessDB V2 verwendet, sind diese Funktionen völlig transparent.

Die Kommunikation zwischen Anwender DLL (Client) und ODK-RTDLL (Server) wird über Interprocess Communication (IPC) abgewickelt.

Es werden Events und ein Shared Memory Bereich benötigt. Die Kommunikationsrichtung ist dabei immer eindeutig. Entweder schreibt der Client auf ein Objekt und liest der Server oder umgekehrt.

Die Struktur ist in der Abbildung 3-2 IPC zwischen Anwender-DLL und ODK-RTDLL, S. 12 dargestellt.

Abbildung 3-2 IPC zwischen Anwender-DLL und ODK-RTDLL



Im Shared Memory gibt es vier Bereiche:

- Request Header
- Request Data
- Response Block
- Response Data

Request Block

Der Requestblock wird nur vom Client beschrieben. Nachdem der Client die gewünschten Daten eingetragen hat, setzt er das **Request-Event**.

Handelt es sich um einen Write-Auftrag, wird im Bereich „Request Data“ die Information gespeichert.

Da in einer WinAC ein DB maximal 65 KB groß sein kann, ist auch dieser Bereich 65 KB groß.

Response Block

Der Response-Block wird nur vom Server beschrieben. Sieht der Server das **Request-Event**, liest er den Request-Block, bearbeitet den Request und schreibt die Antwort in den Response-Block. Anschließend setzt er das **Response-Event**.

Handelt es sich um einen Read-Auftrag, wird im „Response Data“ Bereich die gelesene Information gespeichert.

Da in einer WinAC ein DB maximal 65 KB groß sein kann, ist auch dieser Bereich 65 KB groß.

3.6 Performance

Mit dem Referenz-System wurde Performance des Treibers untersucht.

Microbox IPC427C

Intel Core2Duo U9300 @ 1,20 GHz

2 GB RAM

Windows Embedded Standard 2010 (Windows 7 based) - Microbox Image

WinAC RTX 2010

Tuning-Panel: Mindestruhezeit: 0.0 ms

Mindestzykluszeit: 10.0 ms

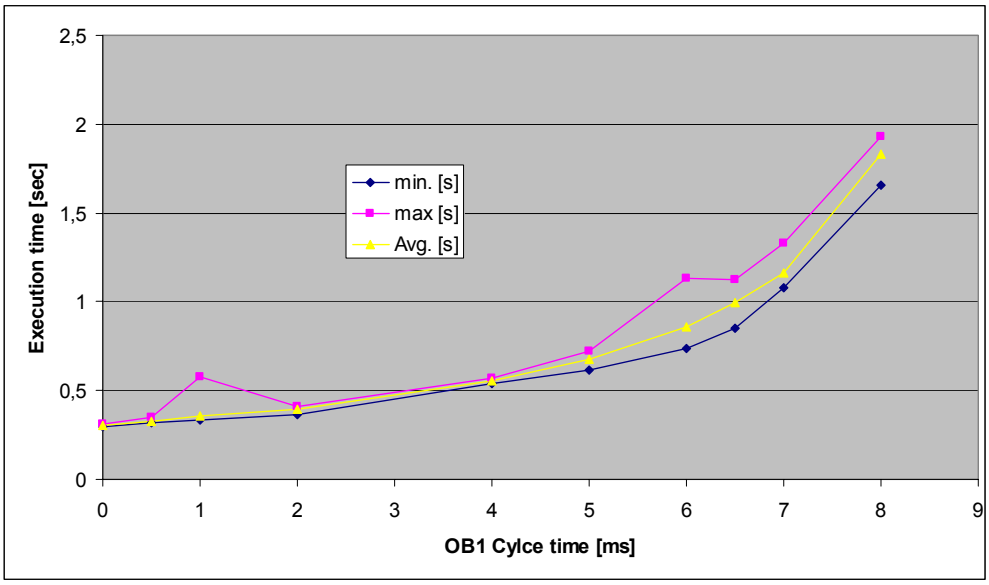
Über eine „Lastsimulation“ (mehrfache Float-Multiplikation in einer Schleife im OB1) wird die Ausführungszeit des OB1 verändert.

Von der Hochsprachenapplikation werden 5.000 Schreib-Befehle über AccessDB V2 abgesetzt. Gemessen wird die Zeit in der Hochsprachenapplikation, bis die 5.000 Befehle abgearbeitet sind. Diese Messung wurde 100x wiederholt.

Tabelle 3-1 Performance 5.000 Schreibbefehle mit AccessDB V2

OB1 Exec time [ms]	min. [s]	max [s]	Avg. [s]
0	0,2969	0,3129	0,3049
0,5	0,3157	0,3509	0,3277
1	0,3322	0,5779	0,3544
2	0,3678	0,4092	0,3927
4	0,5359	0,5704	0,5542
5	0,616	0,7248	0,6777
6	0,7389	1,1292	0,8589
6,5	0,8539	1,128	0,9925
7	1,0755	1,3317	1,1612
8	1,6583	1,9268	1,8289

Abbildung 3-3 Grafik zur Darstellung der Performance von AccessDB V2



Die Geschwindigkeit, mit der über AccessDB V2 Variablen geschrieben werden können, hängt stark von der Auslastung der WinAC ab.

Da die externe Applikation eine Windows-Applikation ist, schwankt die Ausführungszeit der Schreibaufträge.

Lese-Befehle wurden in der selben Geschwindigkeit ausgeführt.

4 Die Step7 Schnittstelle (WinAC FBs)

Um auf die Funktionen von **WinAC AccessDB V2** nutzen zu können, ist nur ein einziger Step7 Funktionsbaustein notwendig.

FB10.001 – FB_INIT_ACCESS_DB

Die Nummer des FBs/DBs im Beispielprojekt können durch den Anwender verändert werden.

4.1 FB_INIT_ACCESS_DB

Mit diesem FB wird die Funktionalität des Treibers initialisiert. Er muss einmalig aufgerufen werden.

Tabelle 4-1 Parameter des FBs FB_INIT_ACCESS_DB

Parameter	In/ Out	Typ	Beschreibung
ERROR	Out	Bool	Fehler aufgetreten
STATUS	Out	WORD	Fehler Information

Zusätzliche Informationen im Instanz-DB von FB_INIT_ACCESS_DB

Neben dem Fehlerbit und dem Fehlercode in der Schnittstelle des FBs stehen dem Anwender im Instanz-DB weitere Informationen zur Verfügung:

Tabelle 4-2 Informationen im Instanz-DB von FB_INIT_ACCESS_DB

Name	In/ Out	Bemerkung
CIF.DLL_VERSION	In	Version der Treiber-DLL

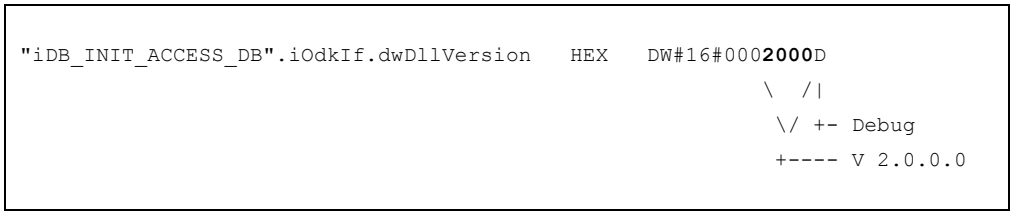
Kodierung der DLL-Version

Die Version der DLL wird hexadezimal kodiert. Die letzte Stelle des DWORDs wird benutzt, um eine Debug- bzw. Release-Version zu kennzeichnen:

D – Debug-Version

A – Release-Version

Abbildung 4-1 Beispiele für Version DLL im Instanz-DB



```
"iDB_INIT_ACCESS_DB".iOdkIf.dwDllVersion   HEX   DW#16#000 2100 A
                                                          \  /|
                                                           \/ +- Release
                                                            +---- V 2.1.0.0
```

Hinweis Die Daten im Instanz-DB (z.B. Softwareversion) sind erst nach einem fehlerfreien Aufruf des INIT-FBs gültig.

5 Die Anwender-Schnittstelle (API)

5.1 Initialisierung

Vor dem ersten Zugriff muss der Treiber initialisiert werden.

DWORD WINAPI InitAccessDB( )		Die Access-DB Applikation initialisieren.

ACHTUNG **Diese Funktion muss genau einmal vor der Benutzung von Schreib- oder Lesefunktionen aufgerufen werden.**

5.1 Aktivieren und Deaktivieren des „Safe Mode“

Der sogenannte „Safe Mode“ wurde für die alte Version von AccessDB V1.x eingeführt, damit der Treiber auch dann richtige Werte liefert, wenn DBs zur Laufzeit neu in die WinAC geladen werden.

Hinweis Diese Funktion ist in der aktuellen Version **nicht** mehr nötig. Die Funktion ist nur aus Gründen der Kompatibilität noch in der API enthalten. Sie ist aber ohne Funktion.

DWORD WINAPI SetSafeMode( )		*Ohne Funktion (Kompatibilität)*

5.2 Funktionen zum Lesen von Daten (WinAC → Applikation)

Hinweis

Für alle Funktionen kann entweder die Nummer eines DB vorgegeben werden oder ein spezieller Code:

66000 – Prozessabbild Eingänge
66001 – Prozessabbild Ausgänge
66002 – Merker

DWORD WINAPI **ReadBOOL**( )		Einen BOOL-Wert von der WinAC lesen *Parameter:* DWORD **DBNummer** (zu lesender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu lesenden Adresse) DWORD **BitOffset** (Bitoffset der zu lesenden Adresse) BIT16 **pValue** (Adresse des zu lesenden Datums)
DWORD WINAPI **ReadBYTE**( )		Einen BYTE-Wert von der WinAC lesen *Parameter:* DWORD **DBNummer** (zu lesender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu lesenden Adresse) BIT8 **pValue** (Adresse des zu lesenden Datums)
DWORD WINAPI **ReadWORD**( )		Einen WORD-Wert von der WinAC lesen *Parameter:* DWORD **DBNummer** (zu lesender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu lesenden Adresse) BIT16 **pValue** (Adresse des zu lesenden Datums)
DWORD WINAPI **ReadDWORD**( )		Einen DWORD-Wert von der WinAC lesen *Parameter:* DWORD **DBNummer** (zu lesender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu lesenden Adresse) BIT32 **pValue** (Adresse des zu lesenden Datums)
DWORD WINAPI **ReadINT**( )		Einen INT-Wert von der WinAC lesen *Parameter:* DWORD **DBNummer** (zu lesender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu lesenden Adresse) SINT16 **pValue** (Adresse des zu lesenden Datums)
DWORD WINAPI **ReadDINT**( )		Einen DINT-Wert von der WinAC lesen *Parameter:* DWORD **DBNummer** (zu lesender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu lesenden Adresse) SINT32 **pValue** (Adresse des zu lesenden Datums)

DWORD WINAPI **ReadREAL**( )		Einen REAL-Wert von der WinAC lesen *Parameter:* DWORD **DBNummer** (zu lesender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu lesenden Adresse) FLOAT **pValue** (Adresse des zu lesenden Datums)
DWORD WINAPI **ReadDATE**( )		Einen DATE-Wert von der WinAC lesen *Parameter:* DWORD **DBNummer** (zu lesender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu lesenden Adresse) UINT16 **pValue** (Adresse des zu lesenden Datums) **ACHTUNG!** Diese Funktion führt keine Konvertierung durch. Sie überträgt die 2 Byte der S7 Variablen vom Type DATE.
DWORD WINAPI **ReadS5TIME**( )		Einen S5TIME-Wert von der WinAC lesen *Parameter:* DWORD **DBNummer** (zu lesender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu lesenden Adresse) BIT16 **pValue** (Adresse des zu lesenden Datums) **ACHTUNG!** Diese Funktion führt keine Konvertierung durch. Sie überträgt die 2 Byte der S7 Variablen vom Type S5TIME.
DWORD WINAPI **ReadS7TIME**( )		Einen S7TIME-Wert von der WinAC lesen *Parameter:* DWORD **DBNummer** (zu lesender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu lesenden Adresse) SINT32 **pValue** (Adresse des zu lesenden Datums) **ACHTUNG!** Diese Funktion führt keine Konvertierung durch. Sie überträgt die 4 Byte der S7 Variablen vom Type TIME.
DWORD WINAPI **ReadS7TIME_OF_DAY**		Einen S7TIME_OF_DAY-Wert von der WinAC lesen *Parameter:* DWORD **DBNummer** (zu lesender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu lesenden Adresse) UINT32 **pValue** (Adresse des zu lesenden Datums) **ACHTUNG!** Diese Funktion führt keine Konvertierung durch. Sie überträgt die 4 Byte der S7 Variablen vom Type TIME_OF_DAY.
DWORD WINAPI **ReadS7STRING( )**		Einen S7STRING von der WinAC lesen *Parameter:* DWORD **DBNummer** (zu lesender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu lesenden Adresse) BIT8 Writemax (Länge des STRINGS in Charactern)

		CHAR **pValue** (Adresse des zu lesenden Strings) **ACHTUNG!** Diese Funktion ermittelt die zu übertragende Anzahl von Bytes aus dem Header des S7-Strings auf der WinAC Seite. Es wird die Anzahl von Bytes gelesen, die in der aktuellen Länge des S7-String gespeichert ist.
DWORD WINAPI **ReadS7STRING_LEN**		Die Längenparameter eines S7-STRING von der WinAC lesen (max. Länge / aktuelle Länge) *Parameter:* DWORD **DBNummer** (zu lesender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu lesenden Adresse) BIT8 ***pMaxLen** (max. Länge des S7-STRING) BIT8 ***pCurLen** (aktuelle Länge des S7-STRING)
DWORD WINAPI **ReadS7BLOCK( )**		Einen S7BLOCK (Array of Byte) von der WinAC lesen *Parameter:* DWORD **DBNummer** (zu lesender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu lesenden Adresse) DWORD **Blocklength** (Länge des Blockes in Bytes) BIT8 **pBlock** (Adresse des zu lesenden Datums)

5.3 Funktionen zum Schreiben von Daten (Applikation → WinAC)

Hinweis

Für alle Funktionen kann entweder die Nummer eines DB vorgegeben werden oder ein spezieller Code:

66000 – Prozessabbild Eingänge
66001 – Prozessabbild Ausgänge
66002 – Merker

DWORD WINAPI **WriteBOOL( )**		Einen BOOL-Wert zur WinAC übertragen *Parameter:* DWORD **DBNummer** (zu beschreibender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu beschreibenden Adresse) DWORD **BitOffset** (Bitoffset der zu beschreibenden Adresse) BIT16 **Value** (zu schreibender Wert)
DWORD WINAPI **WriteBYTE( )**		Einen BYTE-Wert zur WinAC übertragen *Parameter:* DWORD **DBNummer** (zu beschreibender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu beschreibenden Adresse) BIT8 **Value** (zu schreibender Wert)
DWORD WINAPI **WriteWORD( )**		Einen WORD-Wert zur WinAC übertragen *Parameter:* DWORD **DBNummer** (zu beschreibender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu beschreibenden Adresse) BIT16 **Value** (zu schreibender Wert)
DWORD WINAPI **WriteDWORD( )**		Einen DWORD-Wert zur WinAC übertragen *Parameter:* DWORD **DBNummer** (zu beschreibender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu beschreibenden Adresse) BIT32 **Value** (zu schreibender Wert)
DWORD WINAPI **WriteINT( )**		Einen INT-Wert zur WinAC übertragen *Parameter:* DWORD **DBNummer** (zu beschreibender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu beschreibenden Adresse) SINT16 **Value** (zu schreibender Wert)
DWORD WINAPI **WriteDINT( )**		Einen DINT-Wert zur WinAC übertragen *Parameter:* DWORD **DBNummer** (zu beschreibender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu beschreibenden Adresse) SINT32 **Value** (zu schreibender Wert)

DWORD WINAPI **WriteREAL( )**		Einen REAL-Wert zur WinAC übertragen *Parameter:* DWORD **DBNummer** (zu beschreibender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu beschreibenden Adresse) FLOAT **Value** (zu schreibender Wert)
DWORD WINAPI **WriteDATE( )**		Einen DATE-Wert zur WinAC übertragen *Parameter:* DWORD **DBNummer** (zu beschreibender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu beschreibenden Adresse) UINT16 **Value** (zu schreibender Wert) **ACHTUNG!** Diese Funktion führt keine Konvertierung durch. Sie überträgt 2 Byte in die S7 Variablen vom Type DATE.
DWORD WINAPI **WriteS5TIME( )**		Einen S5TIME-Wert zur WinAC übertragen *Parameter:* DWORD **DBNummer** (zu beschreibender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu beschreibenden Adresse) BIT16 **Value** (zu schreibender Wert) **ACHTUNG!** Diese Funktion führt keine Konvertierung durch. Sie überträgt 2 Byte in die S7 Variablen vom Type S5TIME.
DWORD WINAPI **WriteS7TIME( )**		Einen S7TIME-Wert zur WinAC übertragen *Parameter:* DWORD **DBNummer** (zu beschreibender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu beschreibenden Adresse) SINT32 **Value** (zu schreibender Wert) **ACHTUNG!** Diese Funktion führt keine Konvertierung durch. Sie überträgt 4 Byte in die S7 Variablen vom Type TIME.
DWORD WINAPI **WriteS7TIME_OF_DAY**		Einen S7TIME_OF_DAY-Wert zur WinAC übertragen *Parameter:* DWORD **DBNummer** (zu beschreibender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu beschreibenden Adresse) UINT32 **Value** (zu schreibender Wert) **ACHTUNG!** Diese Funktion führt keine Konvertierung durch. Sie überträgt 4 Byte in die S7 Variablen vom Type TIME_OF_DAY.
DWORD WINAPI **WriteS7STRING( )**		Einen S7-STRING zur WinAC übertragen *Parameter:* DWORD **DBNummer** (zu beschreibender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu beschreibenden Adresse)

		BIT8 Writemax (Länge des STRINGS in Charactern) CHAR **pValue** (Adresse des zu schreibenden Strings) **ACHTUNG!** Diese Funktion ermittelt zunächst die maximal mögliche Länge des S7-Strings aus dem Header des S7-Strings. D.h. der Wert „maximale Länge“ des S7-Strings muss korrekt gesetzt sein, wenn diese Funktion verwendet werden soll. **ACHTUNG!** Ist der übergebene String länger als die maximale Länge des S7-Strings, wird nur die maximal mögliche Zahl von Zeichen an die WinAC übertragen.
DWORD WINAPI **WriteS7BLOCK( )**		Einen S7BLOCK (Array of Byte) zur WinAC übertragen *Parameter:* DWORD **DBNummer** (zu beschreibender Datenbaustein) DWORD **ByteOffset** (Byteoffset der zu beschreibenden Adresse) DWORD **Blocklength** (Länge des Blockes in Bytes) BIT8 **pBlock** (Adresse des zu schreibenden Datums)

6 Installation

6.1 Quickstart

Runtime-System

- Treiber mit **setup.bat** installieren (Real-Time DLL wird registriert)
- Die Applikations DLL (**WIcCli.dll**) an einen Pfad kopieren, wo er von der Kundenapplikation gefunden wird (z.B. Applikationspfad oder \system32\)

Engineering-System (Step7 V5.5)

- Step7-Demo-Projekt aus Lieferumfang dearchivieren und öffnen
- Kopieren der Treiber FBs in das eigene Projekt (FB10001, DB10001, SFB65001, SFB65002)
- Funktionsbaustein FB_ACCESS_DB_INIT einmalig aufrufen (z.B. im OB 100)

Engineering-System (TIA Portal V11 / V12)

- Entpacken des Archives mit dem TIA Portal Beispiel
- Öffnen des TIA Projektes in eigener Instanz des TIA Portal
- Kopieren der Treiber FBs in das eigene Projekt (FB10001, DB10001)
- Funktionsbaustein FB_ACCESS_DB_INIT einmalig aufrufen (z.B. im OB 100)

6.2 Installation des WinAC-Treibers auf dem Runtimerechner

Die Installation des WinAC AccessDB beschränkt sich auf das Registrieren der Treiber –RTDLL sowie das Kopieren der Applikations-DLL **WlcCli.dll** in das System32-Verzeichnis. Dazu gibt es eine BATCH-Datei **setup.bat**.

6.2.1 Installation unter Windows XP

Unter Windows XP(e) braucht nur das Batch-File ausgeführt zu werden. Das funktioniert auch direkt von USB Stick.

Hinweis Bitte die Ausgabe des Batch-Files auf Fehlermeldungen prüfen.

6.2.2 Installation unter Windows 7

Unter Windows 7 muss die Batch-Datei mit Administrator-Rechten ausgeführt werden.

Abbildung 6-1 Fehler, wenn setup.bat ohne Administrator Rechte ausgeführt wird

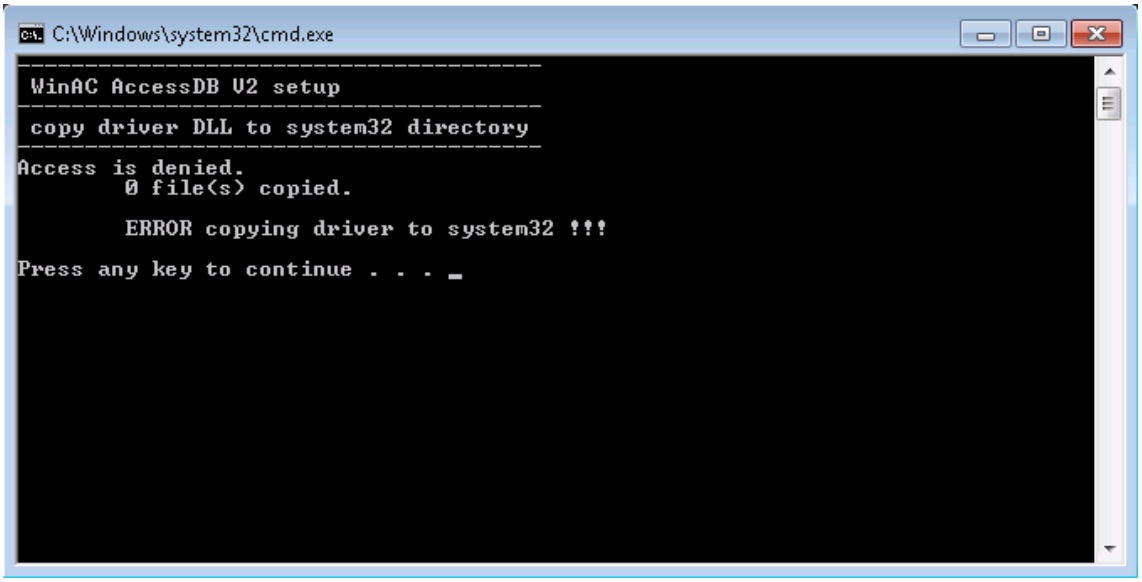
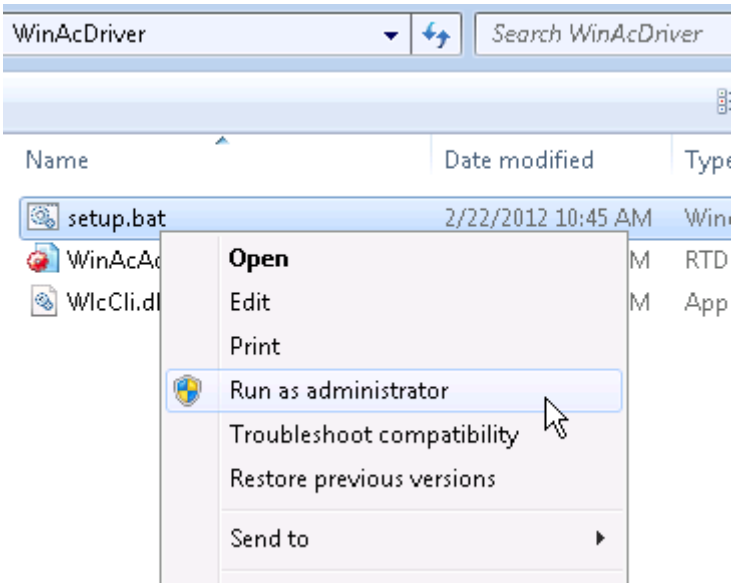


Abbildung 6-2 Ausführen der Setup.bat mit Administrator Rechten



6.3 Installation des WinAC-Treibers auf dem Entwicklungsrechner

Auf dem Entwicklungsrechner werden folgende Komponenten benötigt:

- diese Dokumentation
- das Step7-Beispielprojekt

Hinweis Eine Installation des Treibers (**setup.bat**) auf dem Entwicklungsrechner ist **NICHT** erforderlich.

6.3.1 Verwendung mit Step7 Classic (V5.5)

Dem S7-Demo-Projekt kann der erforderliche FB und der passende Instanz-DB für das Step7 Anwenderprogramm entnommen werden.

Da der Treiber auf dem WinAC ODK beruht, sind außerdem die ODK SFBs SFB 65001 CREA_COM und SFB65002 EXEC_COM zu kopieren.

6.3.2 Verwendung mit TIA Portal V11 SP2 / V12

Im TIA Portal müssen nur der FB FB_ACCESS_DB_INIT sowie der zugehörige Instanz-DB kopiert werden.

Hinweis Die verwendete SIMATIC CPU muss eine WinAC sein. Im anderen Fall (z.B. unspezifische CPU) wird der Treiber-FB fehlerhaft markiert.

7 Anwendungsbeispiele

Das Treiber-Paket enthält ein Step7-Beispielprojekt sowie Beispielapplikationen in verschiedenen Programmiersprachen / Entwicklungsumgebungen.

Die Funktionalität ist die gleiche für Step7 Classic (V5.5) und TIA Portal V11 SP2.

7.1 Verwenden des Step7-Beispielprojekts

Das Step7 Programm zeigt die Verwendung des Treibers.

OB100 Complete Restart

Im Anlauf-OB muss der Treiber gestartet werden (**FB_INIT_ACCESS_DB**).

OB1 CYCL_EXEC

Hier ist keine Aktion notwendig. Deshalb ist der OB leer.

FB10.001 – FB_INIT_ACCESS_DB

Das ist der einzige FB des **WinAC AccessDB V2** Treibers.

DB10.001 – iDB_INIT_ACCESS_DB

Das ist die Instanz-DBs für den FB des **WinAC AccessDB V2** Treibers.

DB6 TEST_DATA

Dieser DB enthält eine Reihe von Variablen, die für die verschiedenen Hochsprachen-Demo-Programme benötigt werden.

7.2 Beispielapplikation VB6

Das VB6 Beispiel zeigt anhand einer kleinen Dialog-basierten Anwendung, wie der **WinAC AccessDB V2** verwendet werden sollte.

Im Modul **ImportFunctions.bas** sind sämtliche Funktionen der Treiber DLL (WlcCli.dll) deklariert. Dieses Modul muss in das VB6 Projekt eingebunden werden, damit die Funktionen verwendet werden können.

Initialisierung

Die Initialisierung des Treibers erfolgt mit dem einmaligen Aufruf von **InitAccessDB()**. Das geschieht in der Funktion **Form_Load()** die immer automatisch beim Starten der Applikation aufgerufen wird.

Zugriff auf WinAC Daten

Nach erfolgreichem Aufruf von InitAccessDB können die Schreib- / Lesefunktionen verwendet werden. Damit können Inhalte von DBs, Merker, Ein-/Ausgänge abgefragt werden.

Safe Mode

Diese Funktionen sind im AccessDB V2 ohne Funktion. Sie sind nur aus Kompatibiliäts-Gründen noch vorhanden.

8 Fehlermeldungen

Der **WinAC AccessDB V2** Treiber kann verschiedene Klassen von Fehlermeldungen liefern:

- Code im FB-Ausgang **STATUS** gemäß WinAC-ODK (siehe Kapitel 8.1 in diesem Dokument)
- Spezielle Fehlermeldungen des ODK-Treibers (siehe Kapitel 8.2 Seite 32 in diesem Dokument)

8.1 Fehlermeldungen vom WinAC ODK

Der Treiber wurde mit dem WinAC ODK (Open Development Kit) entwickelt. Das ODK kann ebenfalls Fehlercodes generieren, die im **STATUS** der FBs zurückgegeben werden.

Tabelle 8-1 WinAC ODK System-Fehlermeldungen

ODK Code (HEX)	Description
0	Success
8001	An exception occurred.
8002	Input: the ANY pointer is invalid.
8003	Input: the ANY pointer range is invalid.
8004	Output: the ANY pointer is invalid.
8005	Output: the ANY pointer range is invalid.
8006	More bytes were written into the output buffer by the extension object than were allocated.
8007	ODK system has not been initialized: no previous call to SFB65001 (CREA_COM).
8008	The supplied handle value does not correspond to a valid extension object.
8009	More bytes were written into the input buffer by the extension object than were allocated.
807F	An internal error occurred.
80C3	Maximum number (32) of parallel jobs/instances exceeded.
8102	The call to CLSIDFromProgID failed.
8103	The call to CoInitializeEx failed.
8104	The call to CoCreateInstance failed.
8105	The library failed to load.
8106	A Windows response timeout occurred.
8107	Controller is in an invalid state for scheduling an OB.
8108	Schedule information for OB is invalid.
8109	Instance ID for SFB65001 call is invalid.
810A	Controller could not load proxy DLL.
810B	The WinAC controller could not create or initialize shared memory area.
810C	Attempt to access unavailable option ocurred.
8201	The Execute command index could not be found

ODK Code (HEX)	Description
8250	No more available positions in the job list
8252	The count is invalid
8253	A data type of an item in the list is invalid
8254	The count specified is invalid
8255	A memory area of an item in the list is invalid
8256	A DB number of an item in the list is invalid
8257	A bit number of an item in the list is invalid
8258	A pBuff of an item in the list is invalid
8259	A data quantity is invalid
825A	The area offset parameter is invalid for this type
825B	The frequency value is invalid
825C	The callback pointer is invalid
825D	The job ID pointer is invalid
825E	The job ID is invalid
825F	Job could not be completed because address is incorrect
8260	Job could not be completed because of protection level
8261	Job could not be completed because of hardware issues
8301	Invalid Thread Execution Priority
8401	Invalid Asynchronous Event
8402	Asynchronous Processor Queue is empty
8403	Asynchronous Processor Queue is full

8.2 Spezielle Fehlermeldungen des WinAC AccessDB V2

Neben dem allgemeinen Fehlerbit der FBs wird im STATUS ein spezieller Fehlercode geliefert, der die Ursache genauer beschreibt.

Tabelle 8-2 Fehlernummern vom WinAC AccessDB V2

0 - no error

Errors with WinAC Handling
0x8501 - error using ODK_Read.. function
0x8502 - error using ODK_Write.. function
0x8503 - can't read init parameter (FB TINIT_AK)
0x8504 - false FB version (does not match with RTDLL version)

Reading / writing S7 strings
0x8531 - error reading STRING - invalid pointer to string
0x8532 - error reading STRING - error reading string len
0x8533 - error reading STRING - string is too large for the STEP 7 string
0x8534 - error reading STRING - string is too large for the output data buffer
0x8535 - error reading STRING - error writing current string len
0x8536 - error reading STRING - error writing max. string len
0x8537 - error reading STRING - error writing string to WinAC

IPC sync errors
0x8841 - error creating shared memory for IPC sync objects
0x8842 - error creating event to real-time side for IPC sync objects
0x8843 - error creating event to Windows side for IPC sync objects
0x8844 - IPC sync object memory undefined
0x8845 - error set event to real-time side
0x8846 - error waiting for command event from Windows side (abandoned)
0x8847 - error waiting for command event from Windows side
0x8848 - error set event to Windows side
0x8849 - error waiting for answer event from real-time side
0x884a - error waiting for answer event from real-time side (abandoned)
0x884b - timeout answer from real-time side - WinAC (driver) is in RUN
0x884c - timeout answer from real-time side - driver never started
0x884d - timeout answer from real-time side - driver still initialising
0x884e - timeout answer from real-time side - WinAC (driver) in STOP
0x884f - timeout answer from real-time side - WinAC (driver) shuts down
0x8850 - timeout waiting for answer event from real-time side - unkown driver state

Handling errors
0x8901 - got no data for WRITE command
0x8902 - data length 0 for WRITE command
0x8903 - got no return variable for READ command
0x8904 - no buffer for return data of READ
0x8905 - data length 0 for READ command
0x8906 - given buffer to small for read data
0x8907 - error reading S7 string - could not determine string length
0x8908 - received command count does not match expected
0x8909 - given DB number not valid
0x890a - IPC version Windows / real-time side does not match!

9 Literaturhinweis

9.1 Literaturangaben

Diese Liste ist keinesfalls vollständig und spiegelt nur eine Auswahl an geeigneter Literatur wieder.

Tabelle 9-1

	Themengebiet	Titel
/1/	STEP7	Automatisieren mit STEP7 in AWL und SCL Hans Berger Publicis MCD Verlag ISBN 3-89578-113-4
/2/	WinAC	Windows Automation Center RTX – WinAC RTX 2010 Betriebsanleitung
/3/		

9.2 Internet-Link-Angaben

Diese Liste ist keinesfalls vollständig und spiegelt nur eine Auswahl an geeigneten Informationen wieder.

Tabelle 9-2

	Themengebiet	Titel
\1\	Referenz auf den Beitrag	http://support.automation.siemens.com/WW/view/de/BeitragsID
\2\	Siemens I IA/DT Customer Support	http://support.automation.siemens.com
\3\	WinAC RTX 2010 Betriebsanleitung	http://support.automation.siemens.com/WW/view/de/43715176
\4\		

10 Historie

Tabelle 10-1 Versionsgeschichte

Version	Datum	Änderung
V2.00	23.01.12	Erste Ausgabe (Vorabstand)
V2.01	29.01.12	- Kleinere Korrekturen - TIA Portal Beispiel ergänzt - Windows 7 Installation beschrieben
V2.02	08.05.13	Getestet mit: - WinAC RTX 2010 (V4.6) - TIA-Portal V12 - WES7 - IPC 427D