

SIEMENS

WinCC

Configuration Manual

Handbuch

Band 1

C79000-G8200-C139-01

Ausgabe Februar 1999

WinCC, SIMATIC, SINEC, STEP sind Marken von Siemens.

Die übrigen Bezeichnungen in diesem Handbuch können Marken sein, deren Benutzung durch Dritte für deren Zwecke die Rechte der Inhaber verletzen können.

(Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhaltes ist nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte vorbehalten, insbesondere für den Fall der Patenterteilung oder GM-Eintragung.)

(Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so daß wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden jedoch regelmäßig überprüft und notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten. Für Verbesserungsvorschläge sind wir dankbar.)

© Siemens AG 1994 - 1999 All rights reserved

Technische Änderungen vorbehalten

C79000-G8200-C139-01
Printed in the Federal Republic of Germany

Siemens Aktiengesellschaft

Inhaltsverzeichnis

1	Projektierungshandbuch	1-1
1.1	Projektierungshandbuch - Hinweise zum Aufbau und zur Verwendung	1-2
2	WinCC - allgemeine Hinweise	2-1
2.1	WinCC - das Konzept	2-2
2.1.1	Die Schnittstellen von WinCC	2-3
2.2	WinCC - Begriffe und ihre Bedeutung	2-6
3	Projektierung - allgemeine Themen.....	3-1
3.1	Vor Projektbeginn.....	3-2
3.2	Festlegungen im einzelnen	3-3
3.2.1	Festlegung: WinCC Projektname.....	3-4
3.2.2	Festlegung: Variablennamen	3-5
3.2.3	Festlegung: Bildnamen.....	3-7
3.2.4	Festlegung: Scripte und Aktionen	3-9
3.2.5	Festlegung: Die Bedienoberfläche	3-10
3.2.6	Festlegung: Das Bedienkonzept	3-16
3.2.7	Festlegung: Die Farbdefinition	3-18
3.2.8	Festlegung: Die Aktualisierungszyklen.....	3-19
3.2.9	Festlegung: Die Benutzerrechte	3-20
3.2.10	Festlegung: Die Alarmierung	3-21
3.2.11	Festlegung: Zur Realisierung.....	3-22
3.3	Besonderheiten bei der Projektierung mit WinCC	3-23
3.3.1	Aktualisierungszyklen - Wie und wo erfolgt die Einstellung.....	3-24
3.3.1.1	Die Aktualisierung im Bild	3-24
3.3.1.2	Arten der Aktualisierungszyklen.....	3-26
3.3.1.3	Bedeutung der Aktualisierungszyklen	3-27
3.3.1.4	Hinweise zur Verwendung der Aktualisierungszyklen.....	3-29
3.3.1.5	Ausführung von Hintergrundscripten (Global Script).....	3-35
3.3.2	WinCC Dynamisierung	3-38
3.3.2.1	Dynamisieren der Eigenschaften	3-38
3.3.2.2	Dynamisieren der Ereignisse	3-39
3.3.2.3	Dynamisierungsarten für Objekte.....	3-39
3.3.3	WinCC Systemumgebung	3-43
3.3.3.1	Verzeichnisstruktur WinCC System	3-43
3.3.4	WinCC Projektumgebung	3-47
3.3.4.1	WinCC Projekt - Verzeichnisstruktur.....	3-47
3.3.5	Automatischer Projektanlauf von WinCC	3-50
3.3.6	Koordiniertes Beenden von WinCC.....	3-53
3.3.7	Datensicherung	3-54
3.3.8	Kopieren eines gesicherten WinCC-Projektes auf eine neue Ziel-Maschine.....	3-56
3.3.9	Wiederverwendung - Übernahme von Projektteilen in ein neues bzw. bestehendes Projekt.....	3-60
3.3.9.1	Übernahme von Bildern.....	3-61
3.3.9.2	Übernahme von Symbolen und Bitmaps	3-63

3.3.9.3	Übernahme einer Projektbibliothek (mit vorprojektierten Symbolen und Anwenderobjekten)	3-65
3.3.9.4	Übernahme von Aktionen	3-68
3.3.9.5	Übernahme von Variablen	3-68
3.3.9.6	Übernahme von mehrsprachigen Texten (aus Bildern, in Meldungen)	3-76
3.3.9.7	Übernahme von Meldungen.....	3-78
3.3.9.8	Übernahme von Meßwerten	3-81
3.3.9.9	Übernahme von Drucklayouts.....	3-83
3.3.9.10	Übernahme von globalen Aktionen	3-83
3.3.9.11	Übernahme von Projektfunktionen.....	3-83
3.3.9.12	Nutzen von Standard-Funktionen	3-83
3.3.9.13	Übernahme der Benutzerverwaltung.....	3-83
3.3.10	Mauslose Bedienung	3-84
3.3.10.1	Bedienung über Tastatur	3-84
3.3.10.2	Bewegung über Bedienobjekte (Eingabefelder und Bedienfelder)	3-89
3.3.10.3	Alarm Logging Funktionstasten zu den Toolbar-Tasten.....	3-91
3.3.10.4	Alarm Logging anlagenspezifisch entworfene Toolbar-Tasten.....	3-95
3.3.10.5	Tag Logging Funktionstasten zu den Toolbar-Tasten.....	3-95
3.3.10.6	Druckauftrag anstoßen	3-98
3.3.10.7	An- oder Abmelden	3-99
3.3.11	Bildbausteintechnik.....	3-100
3.3.11.1	Prozeßbox als Bildbaustein	3-101
3.3.11.2	Bildbaustein mit indirekter Adressierung	3-103
3.3.11.3	Anwenderobjekte.....	3-104
3.3.11.4	Dynamische Instanz	3-107
3.3.11.5	Prototypbilder	3-107
3.3.11.6	OCX-Objekte.....	3-111
3.3.12	Online Projektierung (Runtime) - Hinweise, Einschränkungen.....	3-113
4	WinCC - C-Kurs	4-1
4.1	Die Entwicklungsumgebung für Scripte in WinCC	4-3
4.1.1	Funktionen und Aktionen in WinCC	4-4
4.1.2	Die Editoren für Funktionen und Aktionen.....	4-10
4.1.3	Funktionen und Aktionen erstellen.....	4-13
4.1.4	Funktionen und Aktionen testen	4-15
4.1.5	Funktionen und Aktionen importieren, exportieren	4-17
4.2	Variablen in C.....	4-18
4.2.1	Beispiel 1 - Variablentypen verwenden	4-21
4.2.2	Beispiel 2 - C Variablen in Verbindung mit WinCC Variablen	4-23
4.2.3	Beispiel 3 - Verwendung von Variablen.....	4-25
4.3	Operatoren und mathematische Funktionen in C	4-27
4.3.1	Boolsche Algebra	4-29
4.3.1.1	Logische Grundfunktionen.....	4-29
4.3.2	Beispiel 1 - Operatoren der Grundrechnungsarten verwenden	4-31
4.3.3	Beispiel 2 - mathematische Funktionen.....	4-33
4.3.4	Beispiel 3 - Operatoren für Bitoperationen	4-35
4.4	Zeiger in C	4-37
4.4.1	Beispiel 1 - Zeiger	4-39
4.4.2	Beispiel 2 - Zeiger in Verbindung mit WinCC Variablen.....	4-41
4.4.3	Beispiel 3 - Zeiger in Verbindung mit Stringverarbeitung	4-43

4.5	Schleifen und bedingte Anweisungen in C	4-44
4.5.1	Beispiel 1 - while - Schleife	4-47
4.5.2	Beispiel 2 - do - while - Schleife	4-49
4.5.3	Beispiel 3 - for Schleife	4-51
4.5.4	Beispiel 4- Bedingte Anweisung mit if	4-53
4.5.5	Beispiel 5- Bedingte Anweisung mit switch und case	4-55
4.5.6	Beispiel 6- Verwendung statischer Variablen mit bedingter Anweisung und Rückgabewert.....	4-57
4.5.7	Beispiel 7- Verwendung statischer Variablen mit bedingter Anweisung und Rückgabewert.....	4-58
4.6	Dateioperationen in C	4-59
4.6.1	Beispiel 1 - Datei öffnen, beschreiben und schließen	4-61
4.6.2	Beispiel 2 - Datei öffnen, erweitern und schließen.....	4-63
4.6.3	Beispiel 3 - Datei öffnen, lesen und schließen.....	4-65
4.6.4	Beispiel 4 - Datei löschen	4-67
4.7	Strukturen in C	4-68
4.7.1	Beispiel 1 - Strukturen in C	4-71
4.7.2	Beispiel 2 - C Strukturen in Verbindung mit WinCC.....	4-73
4.8	Global Scripts.....	4-75
4.8.1	Beispiel 1 - Verwenden einer Projektfunktion	4-76
4.8.2	Beispiel 2 - Verwenden von Projektfunktionen, weitere Beispiele .	4-78
4.9	Projekt als Beispiel	4-80
4.10	Einblenden des Sourcecode mit der rechten Maus	4-81
5	Anhang.....	5-1
5.1	Tips und Tricks.....	5-1
5.1.1	Normierte Ein-/Ausgabe am EA-Feld	5-2
5.1.2	Objektspezifische Aktionen bei Bildanwahl	5-3
5.1.3	WinCC-Scope	5-4
5.1.4	Datenbank-Zugriffe.....	5-5
5.1.4.1	Zugriff auf die Datenbank von Excel/MSQuery aus	5-5
5.1.4.2	Zugriff auf die Datenbank von Access aus	5-8
5.1.4.3	Zugriff auf die Datenbank von ISQL aus	5-9
5.1.4.4	Zugriff auf die Datenbank von WinCC-Scope aus	5-10
5.1.4.5	Export aus Datenbank über C-Aktionen	5-12
5.1.4.6	Datenbank-Selektionen	5-13
5.1.5	Serielle Kopplung	5-14
5.1.6	Farbtabelle	5-15
5.2	Dokumentation des S5-Meldesystems.....	5-17
5.2.1	Auflistung der Softwarebausteine	5-18
5.2.2	Hardwarevoraussetzung.....	5-19
5.2.3	Einbinden des S5-Meldesystems in das SIMATIC-S5 Anwenderprogramm.....	5-20
5.2.4	Allgemeine Beschreibung des S5-Meldesystems.....	5-22
5.2.4.1	Aufbau des Offset Datenbausteins	5-23
5.2.4.2	Basismeldungsnummer	5-24
5.2.4.3	Offsetmeldungsnummer / Signalzustände der Meldungen.....	5-24
5.2.4.4	Signalzustandsblock.....	5-26
5.2.4.5	Adresse des letzten Signalzustandsblocks.....	5-26
5.2.4.6	Signalzustände.....	5-27
5.2.4.7	Ruhezustände	5-28

5.2.4.8	Quittierungsbits	5-28
5.2.4.9	Flankenmerker	5-28
5.2.4.10	Aufbau des Parameter Datenbausteins.....	5-28
5.2.4.11	Aufbau eines Meldungsblocks	5-30
5.2.4.12	Meldungsnummer.....	5-31
5.2.4.13	Meldungsstatus	5-31
5.2.4.14	Datum-/ Uhrzeit Stempel	5-31
5.2.4.15	Prozeßvariable	5-31
5.2.4.16	Auftragsnummer / Chargenbezeichnung.....	5-32
5.2.4.17	Reserve	5-32
5.2.4.18	Bildung eines Meldeblocks	5-32
5.2.4.19	Der interne Ringpuffer (FIFO).....	5-32
5.2.4.20	Das Sendefach - Datenübertragung zum überlagerten WinCC-System.....	5-33
5.2.5	Schnittstellenbeschreibung	5-34
5.2.5.1	System Datenbaustein 80.....	5-34
5.2.5.2	Offset Datenbaustein.....	5-34
5.2.5.3	Parameter Datenbaustein.....	5-34
5.2.5.4	Sendefach / Übergabefach	5-35
5.2.6	Parametrierung des S5 Meldesystems / System DB 80	5-36
5.2.7	Projektierungsbeispiel für das S5 Meldesystem	5-41
5.2.7.1	Parametrierung DB 80.....	5-41
5.2.7.2	Einrichten der Datenbausteine.....	5-42
5.2.7.3	Initialisierung der Offset Datenbausteine.....	5-43
5.2.8	Dokumentation der SIMATIC S5 Kommandobausteine	5-46
5.2.8.1	Auflistung der Softwarebausteine	5-46
5.2.8.2	Hardwarevoraussetzung.....	5-47
5.2.8.3	Aufrufparameter von FB 87: EXECUTE	5-47
5.2.9	Schnittstellenbeschreibung	5-48
5.2.9.1	Projektierungsbeispiel für die S5 Kommandobausteine	5-50
5.2.10	Aufgabe und Funktion der S5-Uhrzeitsynchronisation.....	5-51
5.2.10.1	Auflistung der Softwarebausteine	5-51
5.2.10.2	Hardwarevoraussetzung.....	5-51
5.2.11	Aufrufparameter von FB 86 : MELD:UHR	5-52
5.2.12	Datenformate für Datum und Uhrzeit	5-54
5.2.12.1	Uhrendatenbereich CPU 944, CPU 945.....	5-55
5.2.12.2	Uhrendatenbereich CPU 928B, CPU 948	5-56
5.2.12.3	Uhrendatenbereich CPU 946, CPU 947.....	5-57
5.2.12.4	Uhrendatenformat für Meldungsblöcke	5-57
5.2.13	Schnittstellenbeschreibung	5-59
5.2.14	Zusammenspiel mit dem WinCC - Meldesystem	5-60
5.3	Schnittstelle Normierungs-DLL zu AlarmLogging und TagLogging	5-61
5.3.1	Gemeinsame Schnittstelle zu AlarmLogging und TagLogging	5-62
5.3.2	TagLogging-spezifische Zusätze	5-64
5.3.3	API-Funktionen einer WinCC-Normierungs-DLL	5-65
5.3.3.1	Initialisierung der Normierungs-DLL.....	5-65
5.3.3.2	Abfrage der Eigenschaften einer Normierungs-DLL	5-65
5.3.3.3	Abfrage des Namens der Normierungs-DLL	5-68
5.3.4	Shutdown der Normierungs-DLL.....	5-69
5.3.4.1	Erweiterungen der Projektierung.....	5-69
5.3.4.2	Dialogerweiterung bei der Projektierung von S7PMC-Meldungen.	5-70

5.3.4.3	Dialogerweiterung bei der Projektierung von Archivvariablen	5-71
5.3.4.4	Online-Dienste	5-73
5.3.4.5	Registrieren aller Archivvariablen.....	5-74
5.3.4.6	Sprachumschaltung.....	5-76
5.3.5	Normierung	5-77
5.3.5.1	Ableitung von Einzelmeldungen.....	5-77
5.3.5.2	Meldungen quittieren, sperren/freigeben.....	5-78
5.3.5.3	Bearbeitung beim Zustandswechsel	5-79
5.3.5.4	Meldungsupdate der S7PMC-Normierungs-DLL	5-79
5.3.5.5	Normierung von Archivvariablen	5-80
5.3.5.6	Ableitung von einzelnen Archivvariablen-Werten	5-80
5.3.5.7	Sperren / Freigeben von Archivvariablen	5-82
5.3.5.8	Bearbeitung beim Zustandswechsel	5-82
5.4	Erstellung von Bildbausteinen für WinCC.....	5-83
5.5	Globale Bibliothek	5-85
5.5.1	Absperrarmaturen	5-86
5.5.2	Absperrschieber	5-87
5.5.3	Anwender Objekte.....	5-88
5.5.3.1	Anzeigen.....	5-88
5.5.3.2	Bedienpanels	5-88
5.5.3.3	Incr_Decr_Buttons.....	5-88
5.5.3.4	Rohre.....	5-88
5.5.3.5	Tanks.....	5-89
5.5.3.6	Umschalt Buttons	5-90
5.5.3.7	Ventile.....	5-90
5.5.3.8	Zeiger Instrumente	5-90
5.5.4	Button Bilder	5-91
5.5.5	Buttons 3D	5-92
5.5.6	Buttons Sprachumschaltung.....	5-93
5.5.7	DIN30600.....	5-94
5.5.8	E-Symbole	5-95
5.5.9	Fenster.....	5-96
5.5.10	Förderbänder	5-97
5.5.11	ISA Symbole	5-98
5.5.11.1	isa_s55a.....	5-98
5.5.11.2	isa_s55b.....	5-98
5.5.11.3	isa_s55c.....	5-98
5.5.11.4	isa_s55d.....	5-99
5.5.11.5	isa_y32a.....	5-99
5.5.11.6	isa_y32b.....	5-99
5.5.11.7	isa_y32c.....	5-99
5.5.11.8	isa_y32d.....	5-100
5.5.11.9	isa_y32e.....	5-100
5.5.11.10	isa_y32f	5-100
5.5.11.11	isa_y32g.....	5-100
5.5.11.12	isa_y32h.....	5-101
5.5.11.13	isa_y32i.....	5-101
5.5.12	Keyboards.....	5-102
5.5.13	Motoren.....	5-103
5.5.14	Motoren 3D	5-104
5.5.15	PC / PLC.....	5-105

5.5.16	Pumpen	5-106
5.5.17	Regler	5-107
5.5.18	Rohre	5-108
5.5.19	Skalierung	5-109
5.5.20	Textfelder	5-110
5.5.21	Ventile	5-111
5.5.22	Ventile 3D	5-112
5.5.23	Verschiedenes1	5-113
5.5.24	Verschiedenes2	5-114

Vorwort

Zweck des Handbuchs

Dieses Handbuch stellt Ihnen die Möglichkeiten der Projektierung mit WinCC anhand folgender Teile vor:

- ein allgemeiner Teil zu WinCC und zur Projektierung
- eine Einführung in die Scriptverarbeitung
- einen Anhang

Über das Inhaltsverzeichnis oder den Index finden Sie schnell die benötigte Information. Diese steht Ihnen natürlich auch als Onlinedokument im HTML-Format mit erweiterten Suchfunktionen zur Verfügung.

Die Beschreibung der Beispielprojekte zur effektiven und effizienten Projektierung mit WinCC finden Sie im *WinCC Configuration Manual Band 2*.

Voraussetzungen für dieses Handbuch

Kenntnisse in WinCC (**Getting Started**) oder praktische Erfahrung in der Projektierung mit WinCC.

Weitere Unterstützung

Bei technischen Fragen wenden Sie sich bitte an Ihren Siemens-Ansprechpartner, in den für Sie zuständigen Vertretungen und Geschäftsstellen.

Die Adressen finden Sie z.B. im Anhang Siemens weltweit, des Handbuches "Automatisierungssystem S7-300, Aufbau in eine S7-300", Katalogen und in Compuserve (go auforum)...

Darüber hinaus steht Ihnen unsere Hotline unter der Telefonnummer: +49 (911) 895-7000 (Fax 7001) zur Verfügung.

Weitere Informationen finden Sie im Internet unter der Adresse: www.aut.siemens.de/coros/html_00/coros.htm.

Informationen zu SIMATIC-Produkten

Ständig aktuelle Informationen zu den SIMATIC-Produkten erhalten Sie:

- im Internet unter <http://www.aut.siemens.de/>
- über Fax-Polling Nr. 08765-93 02 77 95 00

Darüberhinaus bietet Ihnen der SIMATIC Customer Support Unterstützung durch aktuelle Informationen und Downloads, die beim Einsatz der SIMATIC-Produkte nützlich sein können:

- im Internet unter http://www.aut.siemens.de/support/html_00/index.shtml
- über die SIMATIC Customer Support Mailbox unter der Nummer +49 (911) 895-7100
- Verwenden Sie zur Anwahl der Mailbox ein Modem mit bis zu V.34 (28,8 kBaud), dessen Parameter Sie wie folgt einstellen: 8, N, 1, ANSI, oder wählen Sie sich per ISDN (x.75, 64 kBit) ein.

Den SIMATIC Customer Support erreichen Sie telefonisch unter +49 (911) 895-7000 und per Fax unter +49 (911) 895-7002. Anfragen können Sie auch per Mail im Internet oder per Mail in der o. g. Mailbox stellen.

1 Projektierungshandbuch

Das Projektierungshandbuch ist Bestandteil der WinCC Dokumentation und befaßt sich hauptsächlich mit dem praktischen Einsatz von WinCC in Projekten.

Einleitung

Diese Einleitung vermittelt Ihnen allgemeine Informationen zur Vorgehensweise bei der Realisierung von HMI-Projekten (Human Maschine Interface). Bis vor einigen Jahren wurden HMI-Systeme als Bedien&Beobachtungs Systeme bezeichnet.

In den letzten Jahren sind die Anforderungen an ein System zur Beobachtung und Steuerung von Produktionsvorgängen, sowie für die Archivierung und Weiterverarbeitung der Produktionsdaten, stark gestiegen. Um diesen neuen Anforderungen gerecht zu werden, sind in den letzten Jahren neue HMI-Systeme entwickelt worden.

Eines dieser neuen Systeme ist WinCC. Was die Funktionalität, die Offenheit und die Aktualität (Stand der Softwaretechnik) betrifft, ist WinCC sicher einzigartig.

HMI-Systeme der älteren Generation boten zur Lösung der gestellten Aufgaben oft nur einen Weg. Mit WinCC gibt es für die Realisierung dieser Aufgaben fast immer mehrere Möglichkeiten. Damit Sie stets den besten Lösungsweg im Sinne von Performance und Projektierungsaufwand anwenden, wurde dieses Projektierungshandbuch geschrieben.

Diese Beschreibung soll Ihnen Lösungsvorschläge für den effektiven Einsatz von WinCC in Anlagenprojekten zeigen.

Wir haben die Lösungsvorschläge in WinCC Beispielprojekten realisiert. Diese Beispielprojekte werden mit der WinCC CD geliefert. Sie können diese Lösungsvorschläge direkt in Ihren Projekten verwenden und sparen somit wertvolle Zeit.

1.1 Projektierungshandbuch - Hinweise zum Aufbau und zur Verwendung

Voraussetzung

Vor dem Arbeiten mit diesem Projektierungshandbuch sollten Sie bereits praktische Erfahrungen mit WinCC haben. Für WinCC Neueinsteiger ist das **Getting Started** ein idealer Einstieg. Im Getting Started werden die wichtigsten Themen erklärt und anhand eines kleinen Beispiels praktisch veranschaulicht. Dieses Projektierungshandbuch ist eine Ergänzung zum WinCC Hilfesystem (Online und Dokumentation). Besonderheiten von Objekten, Eigenschaften und anderen Themen sind, soweit nicht in diesem Projektierungshandbuch erklärt, dort nachzulesen.

Hinweis:

Im Kapitel 3.3 *Besonderheiten bei der Projektierung mit WinCC* werden die Projektumgebung und die Dynamisierungsarten von WinCC erklärt.




Inhalt und Aufbau

Die Beschreibung ist in sechs Hauptkapitel gegliedert.

- Der erste Teil ist ein allgemeiner Teil und enthält das Vorwort, die Einleitung und allgemeine Hinweise zu diesem Handbuch.
- Der zweite Teil enthält allgemeine Hinweise zu WinCC
- Der dritte Teil beinhaltet allgemeine und spezielle Hinweise zur strukturierten und effektiven Abwicklung von HMI-Projekten.
- Der vierte Teil beinhaltet einen WinCC C-Kurs. Für Neueinsteiger sind darin die wichtigsten Regeln für den Einsatz der WinCC Scriptsprache enthalten. Der C-Profi findet darin die Besonderheiten der Entwicklungsumgebung.
- Der sechste Teil enthält den Anhang mit Themen aus WinCC *Solutions* und WinCC *Tips & Tricks*, welche nicht in den Beispielprojekten berücksichtigt wurden.

Konventionen


In dieser Beschreibung werden folgende Konventionen verwendet.

Konvention	Beschreibung
Wichtig	Einzelne, wichtige Textpassagen werden fett dargestellt.
<i>Namen</i>	Namen von Dialogboxen, Button und Feldnamen werden kursiv dargestellt.
<i>Eingaben</i>	Eingaben werden kursiv mit blauer Farbe dargestellt.
<i>Menü</i> → <i>Unterpunkt</i> → <i>usw</i>	Anwahl über Menüpunkte werden kursiv mit blauer Farbe dargestellt. Der Pfeil zeigt die Bedienreihenfolge an.
Programm	C-Scripte werden immer in dieser Form dargestellt.
int	In C-Scripten werden Schlüsselworte in blauer Schrift dargestellt.
"Text", 'z'	In C-Scripten werden Zeichenfolgen und einzelne Zeichen in roter Schrift dargestellt.
Kommentar	In C-Scripten ist der Kommentar blaugrün und hat eine kleinere Schriftgröße.
Vom System	Programmteile die das System vorgibt und nicht geändert werden können
	Bedienung mit der linken Maustaste
 R	Bedienung mit der rechten Maustaste
 D	Doppelklick mit der linken Maustaste
Hinweis	Hinweise werden grau unterlegt dargestellt

Wie finden Sie Informationen

Im **Inhaltsverzeichnis** finden Sie die Informationen nach Themen gegliedert.

Im **Index** finden Sie die Informationen nach Stichworten gegliedert .

In der **Online Dokumentation** werden im Register *Suchen* fast alle Wörter (außer Artikel, Bindewörter usw.) in alphabetischer Reihenfolge aufgelistet. Mit  D auf das gesuchte Wort werden alle Kapitel angezeigt, in denen dieses Wort vorkommt.

2 WinCC - allgemeine Hinweise

Der Aufbau, das Konzept und die Arbeitsweise von WinCC wird Ihnen in diesem Abschnitt näher erklärt.

2.1 WinCC - das Konzept

In WinCC gibt es aus Sicht der Projektierung grundsätzlich drei Lösungsansätze:

- die Projektierung mit WinCC Standardmitteln,
- die Verwendung vorhandener Windows Applikationen mit WinCC über DDE, OLE, ODBC und ActiveX,
- die Entwicklung eigener Applikationen (in VisualC++ oder Visual Basic) eingebettet in WinCC.

Für die Einen ist WinCC das HMI-System für die kostengünstige und schnelle Projektierung und für die Anderen eine unbegrenzt ausbaufähige Systemplattform. Durch die Modularität und Flexibilität von WinCC bieten sich bei der Planung und Realisierung von Automatisierungsaufgaben völlig neue Möglichkeiten.

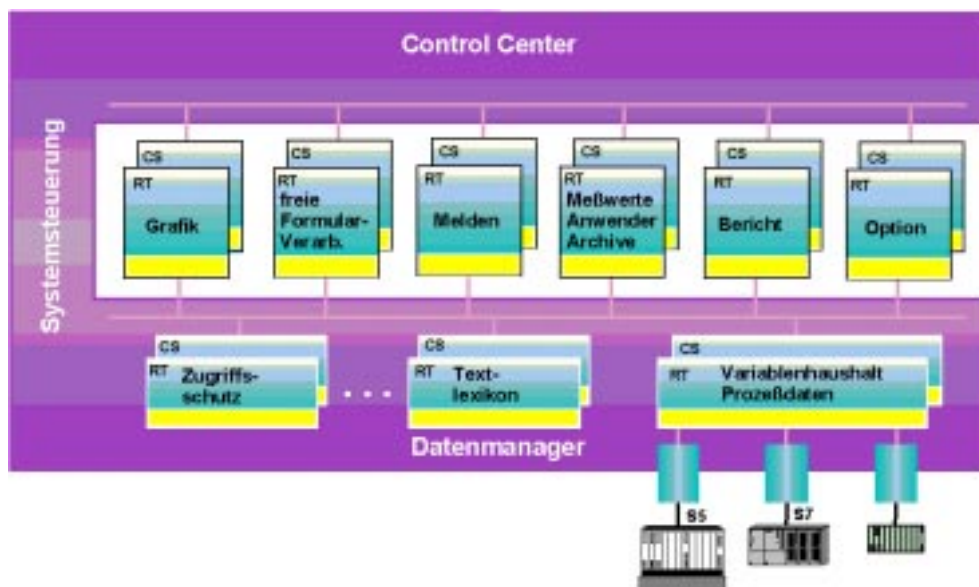
Das Betriebssystem: Basis von WinCC

WinCC basiert auf den 32-Bit Betriebssystemen von Microsoft (derzeit Windows 95 und Windows NT 4.0). Diese Betriebssysteme sind die Standard- Betriebssysteme auf der PC-Plattform.

Der modulare Aufbau von WinCC

WinCC bietet Systemmodule zum Visualisieren, Melden, Erfassen, Archivieren und Protokollieren von Prozeßdaten, sowie zur koordinierten Einbindung von frei formulierten Anwenderrouinen.

Dabei können auch eigene Module eingebunden werden.

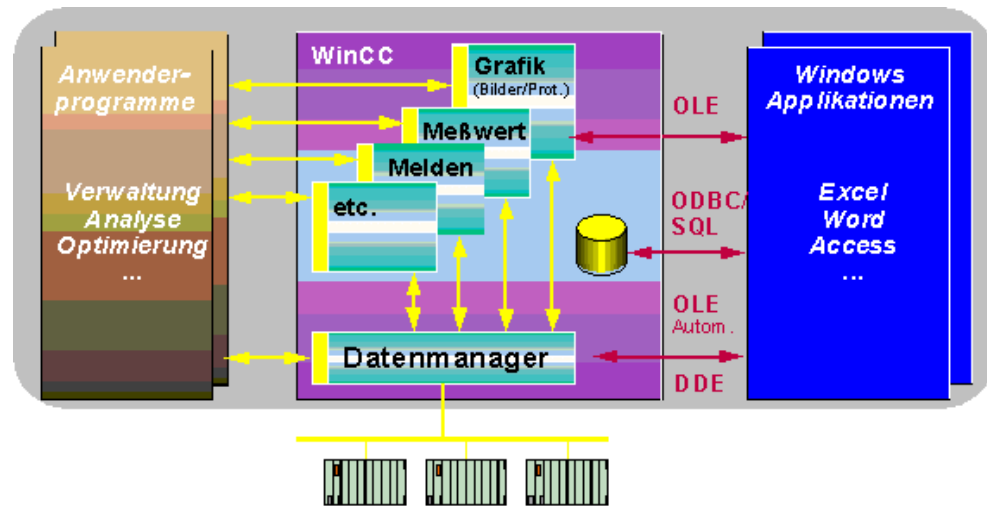


2.1.1 Die Schnittstellen von WinCC

Die Offenheit von WinCC

WinCC ist absolut offen für jede Erweiterung durch den Anwender. Diese absolute Offenheit wird durch den modularen Aufbau von WinCC und seiner leistungsfähigen Programmierschnittstelle erreicht.

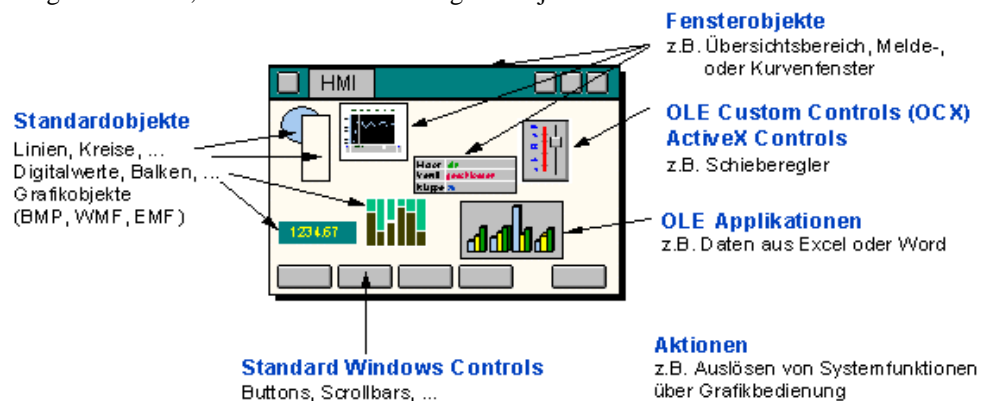
Das nachfolgende Bild zeigt die Möglichkeiten der Anbindung diverser Anwendungen.



Einbinden fremder Applikationen in WinCC

Ganz entscheidend ist, daß WinCC Möglichkeiten bietet, andere Applikationen und Applikationsbausteine **homogen in die Bedienoberfläche für den Prozeßbetrieb zu integrieren**.

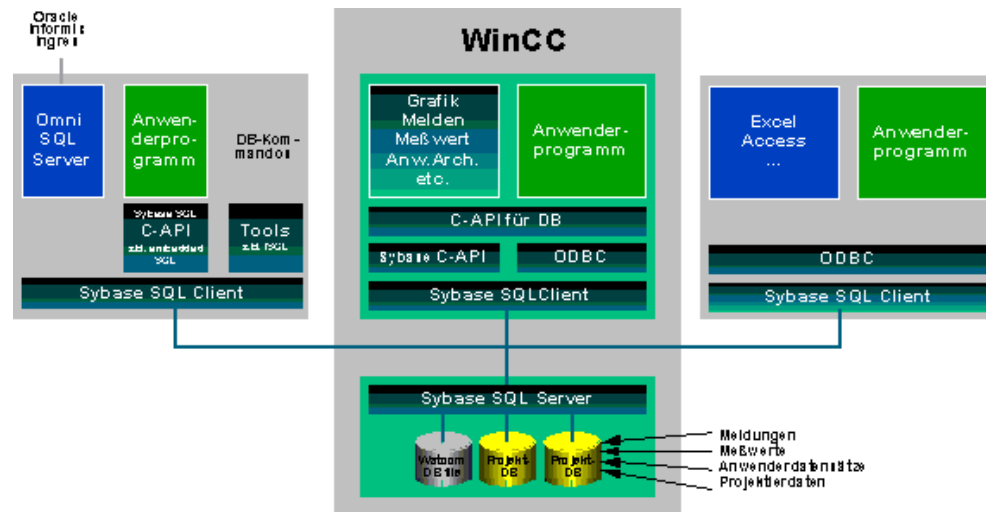
Wie nachfolgend zu erkennen ist, können sowohl OLE Applikationsfenster als auch OLE Custom Controls (32-Bit-OCX-Objekte) bzw. ActiveX Controls in die WinCC-Applikation integriert werden, als wären es WinCC-eigene Objekte.



Die Datenhaltung bei WinCC

Im folgenden Schaubild bildet WinCC den gesamten Mittelteil. Die Grafik zeigt, daß die **Standard-Datenbank Sybase SQL Anywhere** WinCC unterlagert ist. Diese wird benutzt, um (transaktionsgesichert) alle listenorientierten Projektierungsdaten wie Variablenlisten und Meldetexte, aber auch aktuelle Prozeßdaten wie Meldungen, Meßwerte und Anwenderdatensätze abzulegen. Diese Datenbank hat Server-Funktion. WinCC kann über ODBC, aber auch über die offengelegte Programmierschnittstelle (C-API) als Client auf die Datenbank zugreifen.

Das gleiche Recht wird natürlich auch anderen Programmen zugestanden. Aus diesem Grunde hat eine **Windows-Tabellenkalkulation** oder eine **Windows-Datenbank** direkten Zugriff auf den Datenbestand der WinCC-Datenbank, unabhängig davon, ob die Anwendung auf dem selben Rechner oder auf einer vernetzten Station ausgeführt wird. Mit Hilfe der Datenbankabfragesprache SQL und entsprechender Connectivity Tools (z.B. ODBC-Treiber) verfügen aber auch andere Clients (z.B. **UNIX-Datenbanken**, wie Oracle, Informix, Ingres) über eine Zugriffsmöglichkeit auf die WinCC Datenbestände. Der umgekehrte Fall gilt natürlich auch. Insgesamt steht damit einer **Einbindung von WinCC in ein werks- oder unternehmensweites Konzept** nichts im Wege.



2.2 WinCC - Begriffe und ihre Bedeutung

Dieser Abschnitt enthält eine Begriffssammlung rund um das WinCC in alphabetischer Reihenfolge. Viele dieser hier beschriebenen Begriffe sind Ihnen wahrscheinlich bereits bekannt:

HMI	Human Machine Interface
PLC	Programmable Logic Controller
CS	Configuration System
RT	Runtime

3 Projektierung - allgemeine Themen

In diesem Abschnitt finden Sie viele Hinweise und Ideen für die Abwicklung von Projekten mit WinCC. Einige dieser Hinweise sind nicht WinCC spezifisch.

Im Idealfall sollten diese Festlegungen (Regeln für die Projektierung) die Qualität eines Styleguides für die Projektierung und das Design der Runtime Projekte haben.

3.1 Vor Projektbeginn

Bevor Sie mit der Projektierung beginnen, sollten Sie einige Festlegungen und Strukturierungen vornehmen. Dadurch

- vereinfacht sich die Projektierung,
- erhöht sich die Übersichtlichkeit des Projekts,
- vereinfacht sich die Arbeit im Team,
- verbessert sich mitunter die Stabilität und Performance und
- wird die Wartung der Projekte vereinfacht.

Für den Auf- oder Ausbau eines Firmenstandards ist eine klare Festlegung der Aufbaurichtlinien eine Grundvoraussetzung.

Die Festlegungen lassen sich in zwei Klassen teilen:

Festlegungen für die Projektierung

Vor Beginn der Projektierung sollten folgende Festlegungen getroffen werden:

- den Namen für das WinCC-Projekt festlegen.
- die Namen der Variablen festlegen.
- die Namen der WinCC Bilder festlegen.
- Regeln für die Erstellung der Scripte und Aktionen festlegen.
- Regeln für die Projektierung (Firmenstandards, Bibliotheksfunktion, Arbeiten im Team) festlegen.
- Festlegungen für die Dokumentation des Projekts treffen.

Festlegungen für das Runtime Projekt

Festlegungen, die das Runtime Projekt (Ergebnis der Projektierung) betreffen. Diese Festlegungen sind stark vom Einsatzgebiet abhängig (Automobilindustrie, Chemie, Maschinenhersteller, u.a.m.). Es sollten folgende Festlegungen getroffen werden.:

- die Bedienoberfläche (Einteilung des Bildschirms, Schriftart und -Größe, Sprache im Runtime, Darstellung der Objekte) festlegen.
- das Bedienkonzept (Bildhierarchie, Bedienphilosophie, Benutzerrechte, zugelassene Tasten) festlegen.
- die Farbdefinition für Meldungen, Grenzwerte, Zustände, Schrift, u.a.m. festlegen.
- die Kommunikation (Art der Kopplung, Art und Zyklen der Aktualisierung) festlegen.
- das Mengengerüst (Anzahl der Alarmer, Archivwerte, Kurven, Clients, u.a.m.) festlegen.
- Melde- und Archivierungsverfahren festlegen.

3.2 Festlegungen im einzelnen

In diesem Teil des Handbuchs treffen wir Festlegungen, die wir in unseren Beispielprojekten verwenden. Diese Festlegungen sollen Ihnen bei der Erstellung eigener Projekte als Vorlage dienen.

Hinweis:

In unseren Beispielprojekten sind die Namen für Projekte, Bilder, Variablen und Kommentare in den Scripten in englischer Sprache.

Defaultwerte der Projektierwerkzeuge

Bei den meisten Editoren in WinCC können bestimmte Eigenschaften per Defaultwert eingestellt werden. WinCC unterstützt somit Ihren Projektierstil und läßt sich so für bestimmte Aufgabengebiete optimal konfigurieren.

Hinweis:

Als Beispiel sind hier die Möglichkeiten im *Graphics Designer* → *Extras* → *Einstellungen...* genannt. Eine ausführliche Beschreibung zu diesem Thema finden Sie in der Hilfe zum *Graphics Designer*.

3.2.1 Festlegung: WinCC Projektname

Allgemeines

Der Name des WinCC Projekts wird defaultmäßig auch für das Verzeichnis vorgeschlagen, in dem alle Daten abgelegt werden. Der Verzeichnisname kann bei Projekterstellung oder zu einem späteren Zeitpunkt (im Windows Explorer) geändert werden.

Parameter / Grenzen

Es sind alle Zeichen außer gewisse Sonderzeichen (z.B. \ ? ' . ; : /) erlaubt. Weiters sind numerische Werte von 0-9 erlaubt.

Festlegung

In den Beispielprojekten, welche im zweiten Teil des Projektierungshandbuchs beschrieben werden, gilt für den Projektnamen folgende Festlegung:

a...a_nn

Die Bedeutung der Kürzel im einzelnen:

- a Typbezeichnung (a-z, A-Z, keine Sonderzeichen).
- _n Laufende Nummer zur Unterscheidung mehrerer Projekte eines Typs (Zahlen 0-9), Bereich 00-99.

Beispiel: cours_00.mcp, oder pictu_01.mcp

Hinweis für die allgemeine Verwendung

Der WinCC Projektname kann z.B. für die Unterscheidung einzelner Anlagenteile verwendet werden.

Hinweis:

Bei der Rückdokumentation können Sie den Projektnamen in den Ausdrucken mit ausgeben. Die Zuordnung und Auffindbarkeit von Informationen wird dadurch vereinfacht.

3.2.2 Festlegung: Variablennamen

Allgemeines

Die Vergabe von Variablennamen ist nicht mehr auf 8 Zeichen beschränkt. Trotzdem sollte man seine Variablennamen nicht zu lang machen. Die Einhaltung fester Regeln bei der Vergabe der Variablennamen bringt während der Projektierung enorme Vorteile. Bei der Erstellung von WinCC Projekten ist der Aufbau des Variablenhaushalts eine der Schlüsselaufgaben für die schnelle und effektive Projektierung und performante Verarbeitung im Runtime (in Scripten).

Vor der Festlegung der Variablennamen sind einige Besonderheiten der Struktur des Variablenhaushalts in WinCC zu berücksichtigen. Die Erstellung von Gruppen hat nur Auswirkung auf die Darstellung der Variablen im Variablenhaushalt während der Projektierung. Gruppennamen haben keinen Einfluß auf die Eindeutigkeit der Variablennamen. Die Variablennamen innerhalb eines WinCC Projekts müssen eindeutig sein. Die Eindeutigkeit wird vom System überprüft.

Bei der Auswahl von Variablen bietet WinCC viele Hilfen, z.B. bei der Sortierung nach Spalten (Namen, Erstellungsdatum, u.a.m.), oder bei der Verwendung von Filtern. Es kann aber trotzdem sinnvoll sein, wenn der Variablennamen zusätzliche Informationen enthält.

Festlegung

Für Variablennamen in den Beispielprojekten zu diesem Handbuch gilt:

xxxxy_z...z_a...a_nm

Die Bedeutung der Kürzel im einzelnen:

x	Typ
	BIN Binäre Variable
	U08 Vorzeichenloser 8-Bit Wert (<i>unsigned</i>)
	S08 Vorzeichenbehafteter 8-Bit Wert (<i>signed</i>)
	U16 Vorzeichenloser 16-Bit Wert
	S16 Vorzeichenbehafteter 16-Bit Wert
	U32 Vorzeichenloser 32-Bit Wert
	S32 Vorzeichenbehafteter 32-Bit Wert
	G32 Gleitkommazahl 32-Bit IEEE 754
	G64 Gleitkommazahl 64-Bit IEEE 754
	T08 Textvariable 8-Bit Zeichensatz
	T16 Textvariable 16-Bit Zeichensatz
	ROH Rohdatentyp
	TER Textreferenz
	STU Strukturtypen
y	Herkunft
	r reine Lesevariable aus dem PLC (read)
	w Schreib- und Lesevariable aus dem PLC (write)
	i interne Variable in WinCC, ohne Verbindung zum PLC
	x Variable mit indirekter Adressierung (eine Textvariable, deren Inhalt ein Variablenname ist)

- _z Gruppe (entspricht Teilanlagen oder Gebäuden)
_Lack ... z.B. Name der Teilanlage

- _a Variablenname (z.B. Meßstellennamen)
_EU0815V10 ... z.B. Bezeichnung der Meßstelle

- _n Laufende Nummer der Instanz (Zahlen 0-9) Bereich 00-99.

Parameter / Grenzen

Für die Vergabe von Variablennamen bestehen folgende Einschränkungen:

- Das Sonderzeichen @ sollte WinCC Systemvariablen vorbehalten bleiben, die Verwendung dieses Zeichens ist jedoch grundsätzlich möglich.
- Die Sonderzeichen ' und % können nicht verwendet werden.
- Das Sonderzeichen " und die Zeichenfolge // sollen nicht verwendet werden, da diese eine besondere Bedeutung in C-Scripten haben (Einleitung bzw. Abschluß einer Zeichenfolge sowie Einleitung eines Kommentars).
- keine Leerzeichen.
- bei Variablennamen wird nicht zwischen Groß- und Kleinschreibung unterschieden.

Hinweis für die allgemeine Verwendung

Die Vergabe der Variablennamen in den Beispielen soll nur als Vorschlag zu verstehen sein.

Für die Verwendung der Variablen in Scripten und Excel kann es sinnvoll sein, die einzelnen Teile des Variablennamens mit fester Länge (bei Bedarf mit Füllzeichen, 0 oder x) zu vergeben.

Es lassen sich z.B. in Excel große Mengen an Variablen sehr effektiv und einfach erstellen und pflegen. Eine feste Struktur beim Aufbau des Variablennamens erleichtert die Erstellung der Variablenlisten in Excel. Diese in Excel erstellten Variablenlisten können Sie anschließend mit dem Programm

`SmartTools\CC_VariablenImportExport\Var_exim.exe`, welches sich auf Ihrer WinCC CD befindet, in das aktuelle WinCC Projekt importieren.

3.2.3 Festlegung: Bildnamen

Allgemeines

Sollten Bilder in Scripten oder externen Programmen angesprochen werden, ist es sehr hilfreich, bei der Vergabe der Bildnamen eine feste Struktur zu verwenden. Die Länge der Bildnamen sollte man sich ebenfalls genau überlegen. Zu lange Namen (Dateinamen) schränken die Übersichtlichkeit eher ein (Auswahl in Listboxen, Aufrufe in Scripten u.a.m.). In der Praxis hat sich eine Länge von maximal 40 Zeichen bewährt.

Parameter/ Grenzen

Für die Vergabe von Bildnamen bestehen folgende Einschränkungen:

- maximale Länge von 255 Zeichen.
- beliebige Zeichen außer gewissen Sonderzeichen (z.B. / " \ : ? < >).:
- bei Bildnamen wird nicht zwischen Groß- und Kleinschreibung unterschieden.

Festlegung

Für Bildnamen in den Projekten zu diesem Handbuch gilt:

aaaaa_k_x...x_nn

Die Bedeutung der Kürzel im Einzelnen:

- a Bildkennung (a-z, A-Z, keine Sonderzeichen) für die Gruppierung der Bilder.
course... z.B. Name der Bilder im C-Kurs
- _k Bildtyp, Kennung des Bildtyps 0 - 99
 - _0 Startbild
 - _1 Übersichtsbild
 - _2 Tastenbild
 - _3 Anlagenbild
 - _4 Detailbild
 - _5 Meldebild
 - _6 Kurvenbild
 - _7 ...
 - _8 ...
 - _9 Diagnosebilder (nur für Test oder Inbetriebnahme)
- _x Name zur Beschreibung der Bildfunktion (a-z, A-Z, keine Sonderzeichen) maximal 30 Zeichen lang.
_chapter ... z.B. Name der Kapitel im C-Kurs
- _n Laufende Nummer des Typs (Zahlen 0-9) Bereich 0-99.

Hinweis für die allgemeine Verwendung

Die Vergabe der Bildnamen in unseren Beispielen soll nur ein Vorschlag sein. Für die Verwendung eines Teils der mitgelieferten Scripte ist die von uns verwendete Namenskonvention jedoch Voraussetzung.

3.2.4 Festlegung: Scripte und Aktionen

Allgemeines

Sie können in WinCC Projekten eigene Scripte und Aktionen anlegen. Bei der Vergabe der Namen sollten Sie **aussagekräftige Namen** vergeben. Dies ist bei der späteren Verwendung der Scripte sehr hilfreich.

Bei der Projektierung im Global Script (Editor) ist die Verwendung einer Proportionalschriftart eher störend. Sie sollten deshalb zur besseren Lesbarkeit einen Schriftsatz mit konstanter Zeichenbreite verwenden (z.B. Courier).

Die Scripte sollten immer ausreichend kommentiert werden. Der zeitliche Aufwand für das Erstellen der Kommentare steht in keinem Verhältnis zu der Zeit, die man benötigt, um ein schlecht kommentiertes Programm nachzuvollziehen. Obwohl diese Tatsache hinlänglich bekannt ist, wird sie häufig ignoriert.

Festlegung

Für Scripte in den Projekten zu diesem Handbuch gilt:
Wir verwenden die Proportionalschrift *Courier New* in Größe 8,
alle Variablennamen und Kommentare sind in englischer Sprache.

Hinweis für die allgemeine Verwendung

Eine ausführliche Beschreibung für die Verwendung der Scripte, Aktionen und der Editoren finden Sie im Kapitel 4.1 *Die Entwicklungsumgebung für Scripte in WinCC*.

3.2.5 Festlegung: Die Bedienoberfläche

Allgemeines

Es ist von größter Bedeutung, daß bei der Erstellung der Bedienoberfläche mit äußerster Sorgfalt vorgegangen wird. Alle Objekte, die im *Graphics Designer* erstellt werden, erscheinen auf dem Bildschirm im Arbeitsraum des Bedieners.

Die damit erstellten Bilder sind die einzige Schnittstelle zwischen Maschine und Anwender und bedürfen damit besonderer Sorgfalt bei der Erstellung, da sie maßgeblich den Erfolg eines Projekts mitbestimmen. Selbstverständlich ist die Funktion der Anlage wichtiger als das Aussehen des Bildschirms, aber auf lange Sicht können nachlässig erstellte Bilder den Eindruck einer ansonsten gut durchdachten Anlage negativ beeinträchtigen bzw. die Wartungskosten erhöhen.

Diese Bilder sind diejenigen, welche der Anwender (Kunde) täglich sieht.

In einem Bildschirmsystem werden dem Anwender einzig und allein über die Bilder Informationen über den jeweiligen Status der Anlage gegeben. Diese Schnittstelle muß also **möglichst umfassende und leicht verständliche Informationen** zur Verfügung stellen. In WinCC können Sie die Bedienoberfläche frei gestalten. Wie Sie die Bedienoberfläche des Systems gestalten, hängt von der eingesetzten Hardware, von den Anforderungen im Prozeßbetrieb und von bereits vorhandenen Festlegungen ab.

Der Bediener

Bei der Festlegung der Bedienoberfläche sollten Sie den Bediener, für den die Projektierung letzten Endes ausgeführt wird, in den Mittelpunkt Ihrer Betrachtung stellen.

Wenn es gelingt, dem Bediener die erforderlichen Informationen zu geben und diese auf eine übersichtliche Weise darzustellen, wird das Ergebnis eine **höhere Qualität der Produktion und weniger Betriebsunterbrechungen** sein. Die Wartungsarbeiten werden verringert.

Der Bediener braucht so viel Informationen wie möglich. Auf der Grundlage von diesen Daten kann er Entscheidungen treffen, die erforderlich sind, um den Prozeß mit hoher Qualität in Gang zu halten. Der Bediener soll nicht in erster Linie auf Alarmer reagieren (dann ist der Prozeß schon aus dem Gleichgewicht geraten), sondern er soll mit Hilfe seiner Erfahrung, seiner Prozeßkenntnisse und der Informationen vom Bediensystem vorherhersagen können, in welche Richtung sich der Prozeß entwickelt. Der Bediener soll einer Unregelmäßigkeit entgegenwirken können, bevor sie entsteht. Mit WinCC haben Sie die Möglichkeit, dem Bediener diese Informationen effektiv aufzubereiten und anzuzeigen.

Wieviele Informationen sollen in ein Bild?

Für die Entscheidung wieviel Informationen in ein Bild sollten, gibt es zwei Aspekte, die in einem ausgewogenen Verhältnis stehen müssen:

- sind in einem Bild zu viele Informationen enthalten, wird es schwer lesbar und die Suche nach Informationen wird zuviel Zeit erfordern. Die Wahrscheinlichkeit einer Fehlbedienung wird erhöht.
- sind in einem Bild zu wenig Informationen enthalten, wird die Arbeitsbelastung für den Bediener erhöht. Er verliert den Überblick über den Prozeß. Er muß häufig das Bild wechseln, um die benötigte Information zu finden. Verspätete Reaktionen, Bedienungen und Instabilität des zu kontrollierenden Prozesses sind die Folge.

Untersuchungen zeigen, daß ein erfahrener Bediener **so viele Informationen wie möglich in jedem Bild** wünscht, um das Bild nicht so oft wechseln zu müssen. Ein Anfänger dagegen wird bei vielen Informationen in einem Bild verwirrt und verunsichert. Er findet die richtige Information nicht oder nicht rechtzeitig. Aber aus der Erfahrung wissen wir: **Ein Anfänger wird bald erfahren, aber ein Erfahrener wird nie wieder unerfahren.**

Informationen ausblenden

Die Informationen, die gezeigt werden, sollen wichtig und leicht verständlich sein. Sie können **bestimmte Informationen solange ausblenden (z.B. Tag-Nummer, Meßstellenkennzeichen)**, bis sie benötigt werden.

Darstellung der Informationen

Kombinieren Sie bei der Darstellung von Analogwerten Zeigerinstrumente mit Digitalwerten. Für eine schnelle Informationsaufnahme ist die grafische Darstellung (Zeigerinstrumente, Balken ...) von Werten besser geeignet. Um Schwierigkeiten beim Farberkennungsvermögen des Bedieners vorzubeugen (Farbenblindheit), sollten wichtige Veränderungen eines Objekts (Zustand) sowohl durch eine Farb- wie auch durch eine Formänderung angezeigt werden. Wichtige Informationen sollten in einem Bild sofort sichtbar sein. Dies setzt einen gezielten Einsatz von Farbkontrasten voraus.

Farbcodierung

Das menschliche Auge nimmt Farben schneller wahr als z.B. Text. Durch das Arbeiten mit einer sogenannten Farbcodierung bekommt man schneller einen Eindruck davon, in welchem Status (Zustand) sich die verschiedenen Objekte befinden. Es ist wichtig, daß man bei der Farbcodierung, was die Bedeutung der Farben angeht, konsequent vorgeht. Einheitliche Farbfestlegungen in einem Projekt für die Darstellung von Zuständen (z.B. rot für Störung) sind inzwischen Standard. Beim Kunden bereits vorhandene Firmenstandards müssen berücksichtigt werden.

Textdarstellung

Um die Lesbarkeit des Textes zu erhöhen, sollte man sich an einige einfache Regeln halten.

- Die Größe des Textes muß an die Wichtigkeit der Textinformation, aber auch an den zu erwartenden Abstand zwischen dem Bediener und dem Bildschirm angepaßt werden
- Verwenden Sie bevorzugt Kleinbuchstaben. Sie brauchen weniger Platz und sind besser zu lesen als Großbuchstaben, obwohl diese aus der Entfernung besser zu erkennen sind.
- Horizontaler Text ist besser lesbar als vertikaler oder schräggestellter Text.
- Benutzen Sie unterschiedliche Schriftarten für unterschiedliche Informationen (z.B. für Meßstellennamen, Hinweise, u.a.m.).

Das Konzept durchziehen

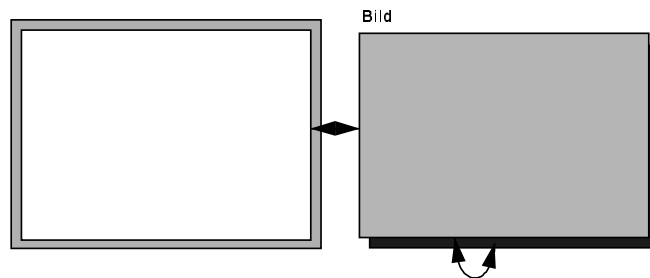
Egal für welches Konzept Sie sich entscheiden, Sie sollten es im gesamten Projekt einhalten. Sie unterstützen damit die intuitive Bedienung der Prozeßbilder. Fehlbedienungen werden unwahrscheinlicher. Dies gilt auch für die verwendeten Objekte. Ein Motor oder eine Pumpe sollte immer gleich aussehen, egal in welchem Bild diese Objekte verwendet werden.

Einteilung des Bildschirms

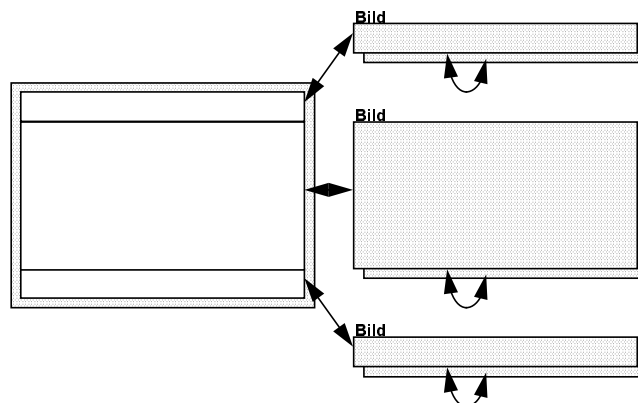
Werden im Prozeß Standard PC-Monitore eingesetzt, so hat sich eine Aufteilung des Bildschirms in drei Bereiche (Übersichts-, Arbeits- und Tastenbereich) als sinnvoll erwiesen.

Läuft Ihre Anwendung jedoch auf einem speziellen Industrie-PC oder Operator Panel mit integrierten Funktionstasten, ist diese Unterteilung nicht immer sinnvoll.

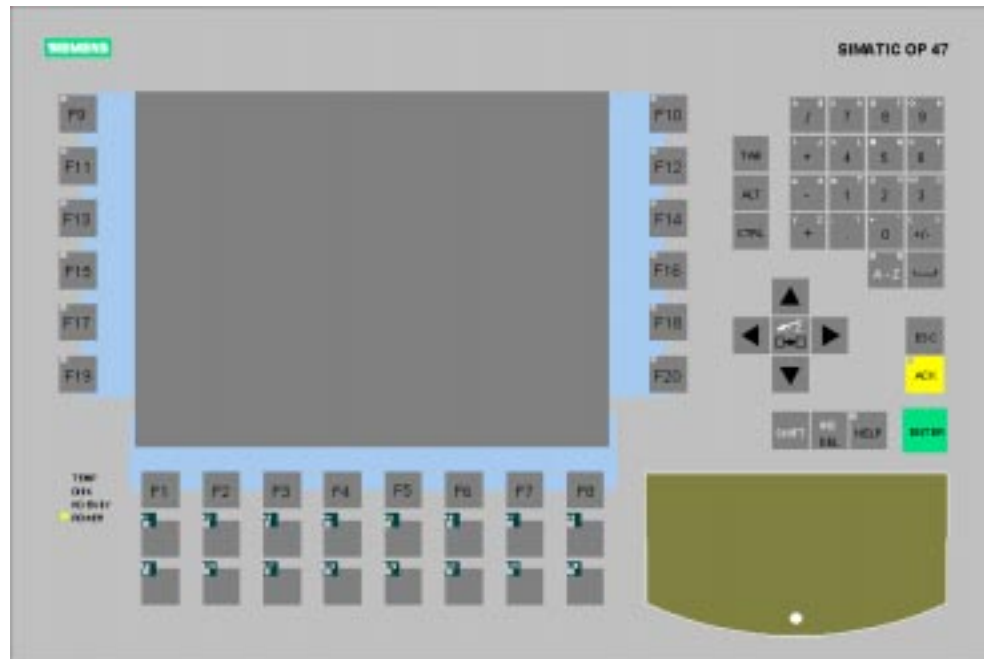
Bilder belegen den gesamten Bildschirmbereich



Der Bildschirm ist in Übersichts-, Tastenbereich und Anlagenbilder aufgeteilt



Beispiel eines Operator Panel



Parameter / Grenzen

Die Größe der einzelnen Bilder kann innerhalb fester Grenzen (min 1 x 1, max. 4096 x 4096 Pixel) frei gewählt werden. Bei Einplatzsystemen mit einem 17" Monitor ist eine maximale Auflösung von 1024 x 768 Pixel zu empfehlen. Bei Mehrplatzsystemen (Multi-VGA) kann eine größere Auflösung sinnvoll sein. Bei Operator Panels beschränkt meist die vorhandene Technik die Auflösung (TFT von 640 x 480 bis 1024 x 768).

Festlegung

Für Bilder in den Projekten zu diesem Handbuch gilt:

Auflösung

Wir verwenden in unseren Projekten Auflösungen von 1024 x 768 und in Ausnahmefällen 800 x 600 Pixel. Für die korrekte Darstellung unserer Beispielprojekte setzen wir eine Einstellung der Farbpalette Ihres PC's von mindestens 65536 Farben voraus.

Texte

Wir verwenden für Meßstellenbezeichnungen die Schriftart Courier, für reine Beschreibungen, für alle anderen Texte und Textdarstellungen die Schriftart Arial. Bei Hinweisboxen im Windows-Stil werden die Schriftarten MS Sans Serif und System verwendet.

Die Schriftgröße passen wir je nach Bedarf an.

Informationen im Bild

Immer wenn es sinnvoll erscheint, blenden wir Informationen aus Bildern aus. Wir zeigen diese Informationen nur bei Bedarf (Bedienung oder automatisch) an.

Wir verwenden in unseren Projekten mehrere unterschiedliche Bildschirmteilungen. Bei vielen bedienbaren Objekten geben wir über Tooltip Hinweise für die Bedienung.

Einteilung des Bildschirms

Wir werden die prinzipiellen Möglichkeiten der Bildschirmteilung projektieren. In den übrigen Projekten jedoch die Aufteilung in Kopfzeile, Arbeitsbereich und Tastenbereich anwenden.

Hinweis für die allgemeine Verwendung

Die Grundaufteilung der verwendeten Konzepte können Sie direkt für Ihre Projekte wiederverwenden.

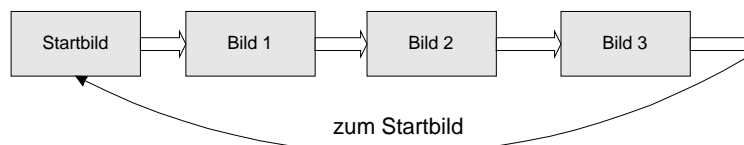
3.2.6 Festlegung: Das Bedienkonzept

Allgemeines

Für die Bedienung Ihrer Anwendung im Prozeß stehen in WinCC die bekannten Eingabemittel wie Tastatur, Maus, Touch Screen oder Industrie Joystick zur Verfügung. Für rauhe Industrieumgebungen, in denen der Einsatz einer Maus nicht möglich ist, können Sie die Bedienung über **Schalt-Cursor** und der **Alpha-Cursor** projektieren. Der Schalt-Cursor springt bedienbare Felder an, der Alpha-Cursor die Eingabefelder. Jede Bedienung läßt sich gegen unberechtigte Zugriffe verriegeln.

Bildanwahl

Das Konzept für die Anwahl der Bilder hängt von mehreren Faktoren ab. Ausschlaggebend dafür ist die Anzahl der Bilder und die Struktur des Prozesses, der dargestellt werden soll. Für kleinere Anwendungen ist die Anordnung der Bilder als Umlaufpuffer möglich.



Bei einer größeren Anzahl von Bildern ist eine hierarchische Anordnung bei der Bildanwahl unerlässlich. Damit sich die Anwahl der Bilder durch das Bedienpersonal möglichst schnell erlernen läßt, ist eine einfache und feste Struktur zu wählen.

Die direkte Anwahl von Bildern ist natürlich auch möglich und bei Kleinstanwendungen (z.B. Kühlhaus) eventuell sinnvoll.

Hierarchie

Eine hierarchische Struktur macht den Prozeß überschaubar, einfach zu handhaben und gibt, falls notwendig, schnell Zugang zu detaillierten Informationen. Eine gängige und häufig eingesetzte Hierarchie besteht aus drei Ebenen.

Ebene 1

In der Ebene 1 werden die Übersichtsbilder eingeordnet. Diese Ebene enthält hauptsächlich Informationen darüber, welche unterschiedlichen Systemabschnitte es im System gibt und wie diese Systemabschnitte zusammenarbeiten. Hier wird ebenfalls gezeigt, ob ein Ereignis (Meldung) in tieferliegenden Ebenen aufgetreten ist.

Ebene 2

In der Ebene 2 werden die Prozeßbilder eingeordnet. Diese Ebene enthält detaillierte Informationen über einen bestimmten Prozeßabschnitt und zeigt, welche Anlagenobjekte zu diesem Prozeßabschnitt gehören. Hier wird ebenfalls gezeigt, auf welches Anlagenobjekt sich ein Alarm bezieht.

Ebene 3

In der Ebene 3 werden die Detailbilder eingeordnet.
Diese Ebene gibt Informationen über einzelne Anlagenobjekte, z.B. Regler, Ventile, Motoren etc. Hier werden Meldungen, Zustände und Prozeßwerte gezeigt. Eventuell werden auch Informationen über das Zusammenwirken mit anderen Anlagenobjekten gegeben.

Festlegung

Für Projekte, die im Zuge der Erstellung dieses Handbuchs entstehen, gilt folgende Festlegung:
Wir werden in unseren Projekten mehrere unterschiedliche Bedienkonzepte verwenden und deren Unterschiede aufzeigen.

Hinweis für die allgemeine Verwendung

Für den Aufbau eines eigenen Bedienkonzeptes können unsere Projekte nur als Anregung dienen. Bei Erweiterungen von Anlagen muß auf bestehende Bedienkonzepte Rücksicht genommen werden. Bei vielen Anwendern gibt es inzwischen Firmenstandards, die bei der Projektierung einzuhalten sind.

Hinweis:

Im WinCC Optionspaket **Basic Process Control** wird ein vorgefertigtes Bedienkonzept angeboten. Das Optionspaket beinhaltet darüber hinaus noch weitere nützliche und leistungsfähige Funktionen (z.B. Storage).

3.2.7 Festlegung: Die Farbdefinition

Allgemeines

Über Farben wird im Umfeld von HMI-Systemen viel und gerne diskutiert. In WinCC können Sie die Farben für Linien, Rahmen, Hintergrund, Füllmuster, und Schrift frei wählen. Es stehen Ihnen alle von Windows unterstützten Farben zur Verfügung. Selbstverständlich lassen sich in WinCC während der Laufzeit die Farben und natürlich auch die anderen Grafikeigenschaften verändern. Für eine kostengünstige Projektierung und eine übersichtliche Darstellung der Prozesse kommt der Farbdefinition eine besondere Bedeutung zu.

Für folgende Bereiche sollte immer eine Definition der Farben durchgeführt werden. Die Festlegung kann eventuell nach DIN EN 60073 entspricht VDE 0199 erfolgen, muß aber immer mit dem Anwender abgestimmt werden:

- Farben für Meldungen (gekommen / gegangen / quittiert)
- Farben für Zustände (ein / aus / gestört)
- Farben für Zeichenobjekte (Leitungen / Füllstände)
- Farben für Warn- und Grenzwerte

Festlegung

Für Farben in den Projekten zu diesem Handbuch gilt:

Für eine korrekte Darstellung der Beispielpunkte ist eine Einstellung größer 256 Farben zu wählen.

In den Beispielpunkten verwenden wir, zur besseren Orientierung, für die einzelnen Themen (Variablen, C-Kurs, Bildprojektierung) jeweils eine eigene Hintergrundfarbe. Im Übersichts- und Tastenbereich ist die Hintergrundfarbe dunkler.

Im Meldesystem ist jeder Meldeklasse und jeder einer Meldeklasse zugeordneten Meldeart ein bestimmter Farbcode zugeordnet.

Hinweis für die allgemeine Verwendung

Nach der Festlegung der Farben sollten Sie gegebenenfalls die Defaulteinstellungen von WinCC anpassen.

Eine Tabelle zur Codierung der Farbwerte in *C-Aktionen* finden Sie im Anhang Kapitel 5.1.6 **Farbtabelle**.

3.2.8 Festlegung: Die Aktualisierungszyklen

Allgemeines

Bei der Festlegung der Aktualisierung ist immer die Betrachtung des Gesamtsystems entscheidend. Was wird aktualisiert und wie oft wird aktualisiert. Die falsche Wahl der Aktualisierungszyklen können negative Einflüsse auf die Performance des HMI-Systems haben.

Bei der Betrachtung eines Gesamtsystems (PLC - Kommunikation - HMI), sollten Änderungen dort erkannt werden, wo sie entstehen, nämlich im Prozeß (PLC). In vielen Fällen stellt das Bussystem den Engpaß für die Datenübertragung dar.

Bei der Festlegung der Aktualisierung von Meßwerten ist darauf zu achten, wie schnell sich der Meßwert tatsächlich ändert. Bei der Temperaturregelung für einen Kessel mit ca. 5.000l Inhalt macht eine Aktualisierung des Istwertes im 500ms Takt absolut keinen Sinn.

32Bit HMI-System

WinCC ist ein reines 32Bit HMI-System, basierend auf Windows 95 und Windows NT. Diese Betriebssysteme sind optimiert für ereignisgesteuerte Bedienung. Wenn Sie diesen Grundsatz bei der Projektierung mit WinCC berücksichtigen, werden Performanceprobleme, auch bei sehr großen Datenmengen, eher die Ausnahme sein.

Festlegung

Für die Aktualisierung in den Projekten zu diesem Handbuch gilt:
Die Aktualisierung erfolgt, soweit es die Aufgabenstellung erlaubt, möglichst ereignisgesteuert. Da wir überwiegend mit *internen Variablen* arbeiten triggern wir oft auf Änderung der Variablen. Bei der Verwendung von externen Variablen kann dies, abhängig vom Prozeßanschluß, zu erhöhter Systembelastung führen. Erlaub die Kommunikation eine ereignisgesteuerte Übertragung, so ist dies für zeitkritische Daten zu wählen. Unkritische Daten können dann vom HMI in angemessenen Zyklen geholt werden (Pollingverfahren).

Hinweis für die allgemeine Verwendung

Eine ausführliche Beschreibung für die Verwendung der Aktualisierungszyklen finden Sie im Kapitel 3.3.1 *Aktualisierungszyklen - Wie und wo erfolgt die Einstellung*.

3.2.9 Festlegung: Die Benutzerrechte

Allgemeines

Im Anlagenbetrieb ist es erforderlich, daß bestimmte Bedienfunktionen vor unberechtigtem Zugriff geschützt werden. Eine weitere Forderung ist, daß nur bestimmte Personen auf das Konfigurationssystem zugreifen dürfen.

Im *User Administrator* können Benutzer und Benutzergruppen angelegt und verschiedene Berechtigungsstufen definiert werden. Diese Berechtigungsstufen können mit den Bedienelementen in den Bildern verbunden werden.

Den Benutzergruppen und Benutzern können individuell verschiedene Berechtigungsstufen zugewiesen werden.

Festlegung

In den Beispielprojekten *course_00* und *varia_00* steht jedem Benutzer die Projektbedienung frei.

Im Beispielprojekt *pictu_00* ist die Bedienung im Projekt erst nach Anmeldung möglich. Das Paßwort entspricht dem Projektnamen (*pictu_00*).

Die Buttons für die Anwahl der einzelnen Themen sind mit der Berechtigungsstufe *Projektbedienung* verbunden.

Hinweis für allgemeine Verwendung

Eine Beschreibung für die Vergabe der Benutzerrechte finden Sie im Configuration Manual Teil 2 im Beispielprojekt *pictu_00* im Kapitel 3.3 WinCC beenden / Bedienfreigaben.

3.2.10 Festlegung: Die Alarmierung

Allgemeines

WinCC unterstützt grundsätzlich zwei Meldeverfahren:

- Das **Bitmeldeverfahren** ist ein universelles Verfahren, das Meldungen aus beliebigen Automatisierungssystemen erlaubt. WinCC überwacht den Flankenwechsel ausgewählter Binärvariablen selbst und leitet daraus Meldungseignisse ab.
- Das **Zeitfolgerichtige Melden** setzt voraus, daß die Automatisierungssysteme die Meldungen selbst bilden und mit Zeitstempel und eventuell mit Prozeßwerten in einem vordefinierten Format an WinCC senden. Die chronologische Ordnung von Meldungen unterschiedlicher Automatisierungssysteme wird erst durch dieses Meldeverfahren möglich. Siehe Kapitel 5.2 *Dokumentation des S5-Meldesystems*.

Was soll gemeldet werden?

Bei der Festlegung, welche Ereignisse und Zustände gemeldet werden, geht man oft den vermeintlich sicheren Weg und meldet alle Ereignisse und Zustandsänderungen. Damit wird dem Bediener die Entscheidung überlassen welche Meldungen er sich zuerst ansieht. Werden in einer Anlage zu viele Ereignisse gemeldet, passiert es erfahrungsgemäß, daß wichtige Meldungen erst zu spät beachtet werden.

Hinweis für allgemeine Verwendung

Die Darstellung der Meldungen und die Auswahl der Meldungen für die Archivierung kann geändert und den eigenen Anforderungen angepaßt werden.

3.2.11 Festlegung: Zur Realisierung

Allgemeines

Für die Realisierung eines Projekts ist es sinnvoll, für die Ablage der Daten eine feste Struktur zu verwenden. Die Festlegung beginnt mit der Entscheidung, auf welchem Laufwerk das WinCC Projekt angelegt wird. Der nächste Schritt betrifft den Aufbau der Verzeichnisstruktur, u.a.m.

Nach unseren Erfahrungen ist es sinnvoll, alle Daten eines Projekts unter einem Verzeichnis mit entsprechenden Unterverzeichnissen abzulegen. Dieses Vorgehen hat Vorteile bei der Projektbearbeitung, aber vor allem bei der Datensicherung.

Hinweis:

Die Ausstattung der PCs ist sehr unterschiedlich. Um sich bei der Vergabe des Ziellaufwerks für die Projektbearbeitung von diesen Abhängigkeiten zu lösen, bieten sich **virtuelle Laufwerke** an. Die Zuordnung des Verzeichnisses zum virtuellen Laufwerk kann jederzeit geändert werden.

Festlegung Verzeichnisse

Neben den Verzeichnissen, die durch WinCC erzeugt werden, legen Sie bei Bedarf weitere Verzeichnisse für Word, Excel und für temporäre Dateien an.

3.3 Besonderheiten bei der Projektierung mit WinCC

In den nachfolgenden Kapiteln werden übergreifende Themen für die Projektierung mit WinCC behandelt.

Diese Themen sind eine Ergänzung zur Hilfe von WinCC.

3.3.1 Aktualisierungszyklen - Wie und wo erfolgt die Einstellung

Die Festlegung der Aktualisierungszyklen ist eine der bedeutendsten Einstellungen im Visualisierungssystem. Aufgrund der Festlegungen werden folgende Eigenschaften beeinflusst:

- der Bildaufbau
- die Aktualisierung der Objekte des aktuell aufgeschlagenen Bildes an der Visualisierungsstation (*Graphics Designer*)
- die Bearbeitung von Hintergrundscripten (*Global Script*)
- die Aktivierung des Datenmanagers sowie die Prozeßkommunikation

Weitere Zeitgrößen werden in der Meßwertverarbeitung (*Tag Logging*) unter den Archivierungszeiten eingestellt.

Datenmanager

Abhängig von den eingestellten Aktualisierungszyklen werden die aktuellen Variablenwerte jeweils vom Datenmanager, dem zentralen Verwalter des Variablenhaushalts, angefordert. Siehe Kapitel 3.3.2 *WinCC Dynamisierung*.

Der Datenmanager ermittelt die neuen Prozeßdaten über die Kommunikationskanäle und versorgt die Applikationen mit diesen aktuellen Werten. Das Anfordern von Daten bedeutet somit jeweils einen Wechsel zwischen verschiedenen Tasks (*Graphics Designer*, *Datenmanager* etc.). Abhängig von der Projektierung kann dies zu sehr unterschiedlichen Systembelastungen führen.

3.3.1.1 Die Aktualisierung im Bild

Bildaktualisierung

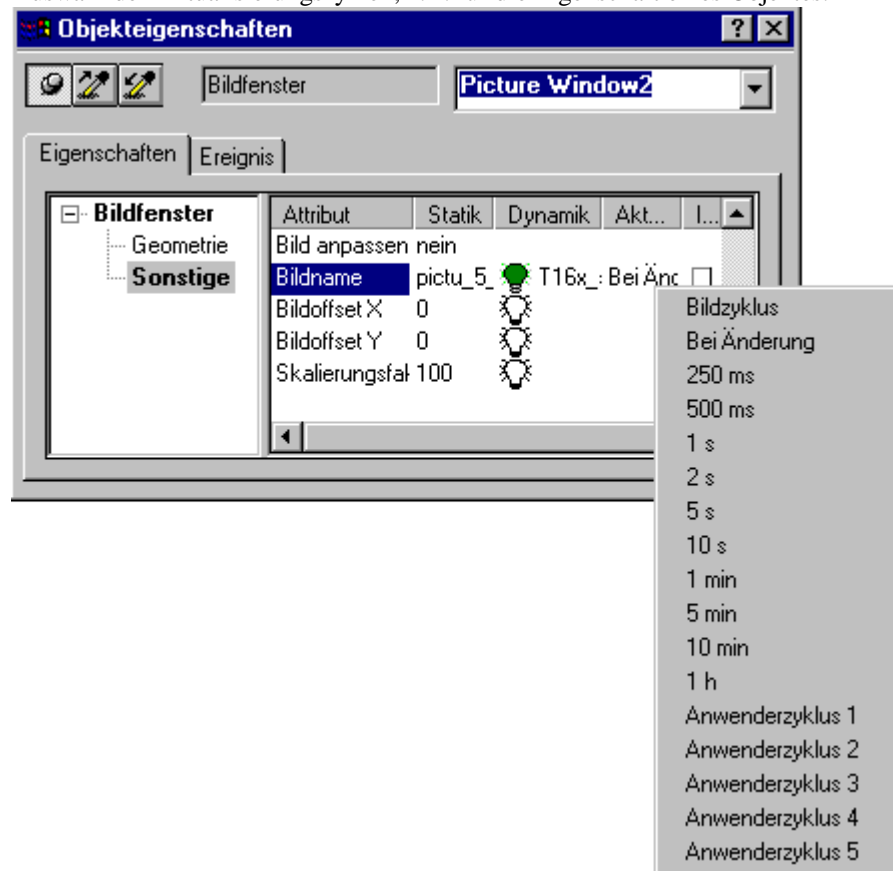
Die Aktualisierung der einzelnen Eigenschaften der Objekte im Bild bezieht sich nach dem Bildaufschlag auf die dynamisierten Objekte. Über den Aktualisierungszyklus wird die Aktualität der Eigenschaften des jeweiligen Objektes im Bild bestimmt. Der Aktualisierungszyklus der dynamisierten Objekte kann bei folgenden Dynamisierungsarten vom Projektteur bzw. vom System eingestellt werden:

Dynamisierungsart	Standard-Einstellung	Projektierungsanpassung
Konfigurationsdialog	Variablentrigger 2 Sek. Oder Ereignistrigger (z.B. Bedienung)	Anpassung der Zeitzyklen
Dynamic-Wizard	Abhängig von der Art der Dynamisierung besteht die Auswahl aus <ul style="list-style-type: none"> • Ereignistrigger, • Zeitzyklus • Variablentrigger 	Anpassung der Zeitzyklen, Ereignisse oder Variablen
<i>Direktverbindung</i>	Ereignistrigger	
Variablenanbindung	Variablentrigger 2 Sek.	Anpassung der Zeitzyklen

Dynamisierungsart	Standard-Einstellung	Projektierungsanpassung
<i>Dynamik-Dialog</i>	Variablentrigger 2 Sek.	Anpassung der Zeitzyklen, Variablentrigger
<i>C-Aktion</i> für Eigenschaften	Zeitzyklus 2 Sek.	Anpassung der Zeitzyklen, Variablentrigger direktes Lesen von der Steuerung
Objekt-Eigenschaft	Einstellung abhängig von der Dynamisierung	Bearbeitung der Spalte Aktualisierungszyklus

Die auszuwählenden Aktualisierungszyklen sind von WinCC vorgegeben und können durch eigene anwenderdefinierte Zeitzyklen ergänzt werden.

Auswahl der Aktualisierungszyklen, z.B. für die Eigenschaft eines Objektes:




3.3.1.2 Arten der Aktualisierungszyklen

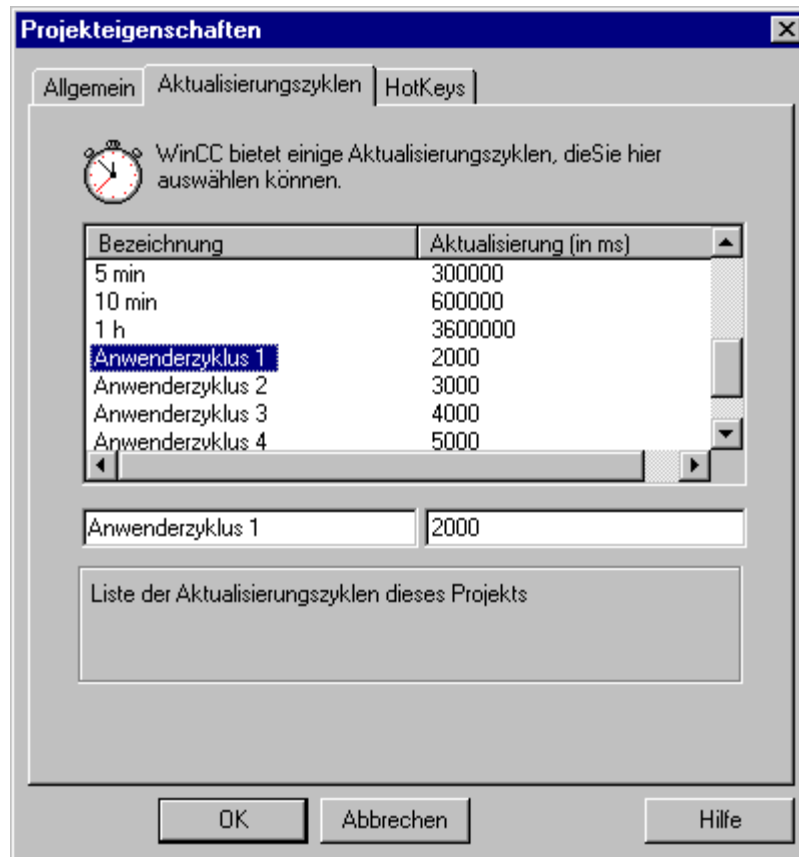
Bei den Aktualisierungszyklen werden folgende Arten unterschieden:

Art	Standard-Einstellung
Standardzyklus	Zeitzyklus von 2 Sekunden
Zeitzyklus	2 Sekunden
Variablentrigger	2 Sekunden
Bildzyklus	2 Sekunden

Art	Standard-Einstellung
Fensterzyklus	bei Änderung
benutzerdefinierte Zeitzyklen	Anwenderzyklus1: 2 Sek Anwenderzyklus2: 3 Sek Anwenderzyklus3: 4 Sek Anwenderzyklus4: 5 Sek Anwenderzyklus5: 10 Sek

Anwenderzyklus

Bis zu 5 Anwenderzyklen können **projektbezogen** definiert werden. Ist im *Control Center* in der linken Baumstruktur der Projektname selektiert, kann über den Button  in der Symbolleiste der Dialog *Projekteigenschaften* geöffnet werden. Im Dialog *Projekteigenschaften* auf der Registerkarte *Aktualisierungszyklen* werden am Ende der Liste der eingestellten Standard-Aktualisierungszyklen die Anwenderzyklen 1...5 für die projektbezogene Definition angeboten. Nur diese Anwenderzyklen können parametrisiert werden.



Über diesen Weg können Zeitzyklen, die noch nicht als Zeiteinheiten zur Verfügung stehen (z.B. 200 ms) definiert werden.

Der Anwenderzyklus kann von 100 ms bis zu 10 Stunden festgelegt werden. Die Bezeichnung für die Anwenderzyklen ist frei wählbar.

Diese projektbezogenen Zeiteinheiten können für ausgewählte Objekte eingesetzt werden, deren Aktualisierungszyklus zu einem späteren Zeitpunkt geändert werden muß. Eine Optimierungsmaßnahme könnte ein Grund für die Änderung der Zeitzyklen sein. Die anwenderdefinierten Aktualisierungszyklen eignen sich dazu, nachträglich an nur **einer** zentralen Stelle den eingestellten Zeitzyklus zu modifizieren. Die einzelnen Objekte der Bilder müssen dann nicht mehr zusätzlich angepaßt werden. Dieser Weg der Definition von Anwenderzyklen sollte daher für **wartungsfreundliche Projekte** in Betracht gezogen werden.

3.3.1.3 Bedeutung der Aktualisierungszyklen

Vor dem Einsatz der möglichen Aktualisierungszyklen muß zunächst die Bedeutung der verschiedenen Aktualisierungszyklen betrachtet werden.

Bei den Aktualisierungszyklen werden folgende Arten unterschieden:

Art	Bedeutung
Standardzyklus	Zeitzyklus
Zeitzyklus	Nach der jeweils eingestellten Zeit wird die Eigenschaft bzw. Aktion des einzelnen Objektes aktualisiert. Dies bedeutet, daß die Variablen jeweils einzel n vom Datenmanager angefordert werden.
Variablentrigger	Abhängig von der eingestellten Zykluszeit werden jeweils nach Ablauf der Zeitspanne die Variablen vom System ermittelt und gegen Wertänderung geprüft. Ändert sich in dem eingestellten Zeitrahmen der Wert mindestens einer ausgewählten Variable, so dient dies als Trigger für die davon abhängigen Eigenschaften bzw. Aktionen. Alle Variablenwerte werden gemeinsam vom Datenmanager angefordert.
Bildzyklus	Aktualisierung der Eigenschaften des aktuellen Bildobjektes sowie aller Objekte, die über den Aktualisierungszyklus Bildzyklus getriggert werden.
Fensterzyklus	Aktualisierung der Eigenschaften des Fensterobjektes sowie aller Objekte die über den Aktualisierungszyklus Fensterzyklus getriggert werden.
benutzerdefinierte Zeitzyklen	Zeiteinheiten, die projektbezogen definiert werden können.
<i>C-Aktion</i> für direktes Lesen von der Steuerung	Mittels interner Funktionen in den <i>C-Aktionen</i> können Werte direkt von der Steuerung gelesen werden. Die weitere Bearbeitung der Folgebefehle in der <i>C-Aktion</i> wird erst nach dem Lesen der Prozeßwerte fortgesetzt (synchrones Lesen).

Hinweis:

Das Anfordern des aktuellen Variablenwertes vom Datenmanager führt jeweils zum Taskwechsel und einem Datenaustausch zwischen den einzelnen Tasks. Zusätzlich müssen die Variablenwerte vom Datenmanager über den Kommunikationskanal der angebotenen Automatisierungsgeräte angefordert werden. Dies erfolgt abhängig von der Art der Kommunikation über Anforderungstelegramme an die Kommunikationsschnittstelle (FETCH) und Datentelegramme von der Automatisierungseinheit zurück an WinCC.

3.3.1.4 Hinweise zur Verwendung der Aktualisierungszyklen


Bei Verwendung der Aktualisierungszyklen wird abhängig von der jeweiligen Art folgende Einstellung empfohlen:

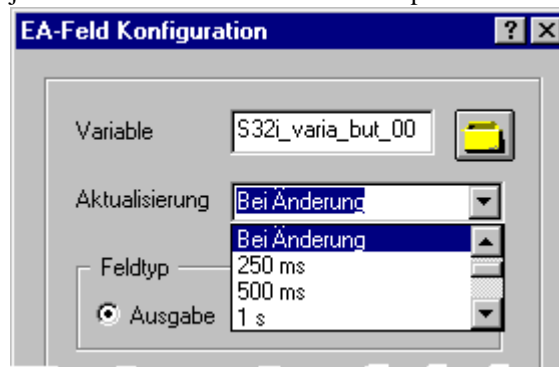
Art	Standard-Einstellung	Empfehlung für die Projektierung
Standardzyklus	Zeitzyklus von 2 Sekunden	<p><i>Dynamik-Dialog</i> oder <i>C-Aktion</i>:</p> <p>falls eine Abhängigkeit von Variablen vorliegt sollte auf jeden Fall eine Variablentriggerung eingesetzt werden. Dies führt zu einer Reduzierung der Taskwechsel sowie der Kommunikation zwischen den Tasks.</p> <p>Die Variablentriggerung <i>bei Änderung</i> darf nur gezielt eingesetzt werden, da dies zu einer höheren Systembelastung führen kann! Die Variablen werden in diesem Fall ständig auf Änderung geprüft. Dieser Polling-Mechanismus führt immer zu einer höheren Systembelastung.</p> <p>Bei Standard-Objekten wird ein Zyklus von 1 bis 2 Sekunden empfohlen.</p>
Zeitzyklus	2 Sekunden	<p>Den Zeitzyklus jeweils vom Objekttyp bzw. der Objekteigenschaft abhängig machen. Die Trägheit von Prozeßkomponenten (Tankfüllungen oder Temperaturen im Gegensatz zu Schalthandlungen) sollte ebenfalls berücksichtigt werden.</p> <p>Bei Standard-Objekten wird ein Zyklus von 1 bis 2 Sekunden empfohlen.</p>
Variablen-trigger	2 Sekunden (bei <i>Dynamik-Dialog</i>)	<p>Falls diese Aktualisierungsmöglichkeit projektierbar (abhängig von der Dynamisierungsart) ist, sollte diese vorzugsweise eingesetzt werden! Berücksichtigen Sie bei einer Abhängigkeit von Variablen auf jeden Fall alle Variablen, die für eine Änderung der Eigenschaft bzw. für den Aktionsdurchlauf verantwortlich sind. Nur die in der Liste aufgeführten Variablen dienen als Trigger für die Aktualisierung der dynamisierten Eigenschaft bzw. Aktion.</p> <p>Die Variablentriggerung <i>bei Änderung</i> sollte nur gezielt eingesetzt werden. Sobald sich eine der ausgewählten Variablen geändert hat, wird der Trigger für diese Eigenschaft bzw. Aktion ausgelöst. Dieser Polling-Mechanismus führt zu einer höheren Systembelastung.</p>
Bildzyklus	2 Sekunden	<p>Dieser sollte nur dann verkleinert werden, wenn sich die dynamisierten Eigenschaften des Bildobjektes selbst in einem kürzeren Zeitraum ändern und damit aktualisiert werden müssen. Eine Erhöhung dieses Bildzyklus verringert die Systembelastung.</p>
Fensterzyklus	bei Änderung	<p>Handelt es sich um ein Bildfenster, das z.B. zum Verstellen von Prozeßgrößen eingeblendet wird (Prozeßbox), so ist diese Einstellung sinnvoll.</p> <p>Wird das Bildfenster für Informationszwecke ständig angezeigt (z.B. Bildschirmgliederung), so sollte die</p>

Art	Standard-Einstellung	Empfehlung für die Projektierung
		Aktualisierung des Fensters sowie der Inhalte auf Variablentrigger oder Zeitzyklus eingestellt werden.
<i>C-Aktion</i> für direktes Lesen von der Steuerung		Die Internen Funktionen (z.B. GetTagWordWait) zum synchronem Lesen von Prozeßwerten (direkt von der Steuerung) sollten nur gezielt eingesetzt werden. Der Einsatz dieser Funktionen erfordert ein Pollen durch das System (Aktionssteuerung) und führt damit zu einer erhöhten Kommunikationsbelastung.


Die folgenden Beispiele zeigen, wo die Aktualisierungszyklen jeweils eingestellt werden:

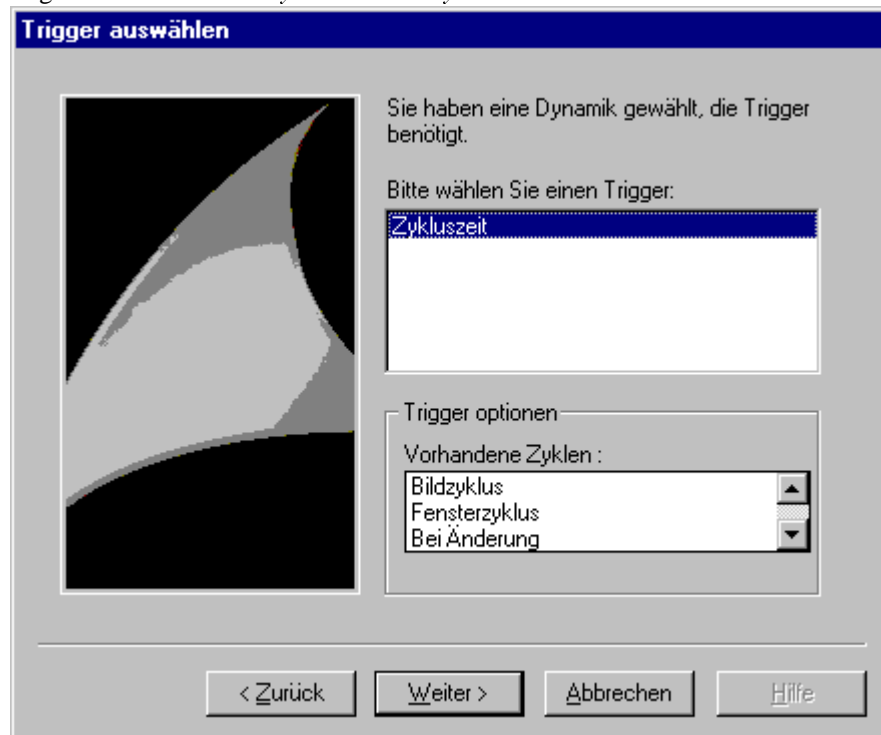
Konfigurationsdialog

Dieser Dialog erscheint bei der Projektierung eines *Smart-Objektes* → *E/A-Feld*. Er kann jedoch auch über  **R** über dem entsprechenden Objekt aufgerufen werden.




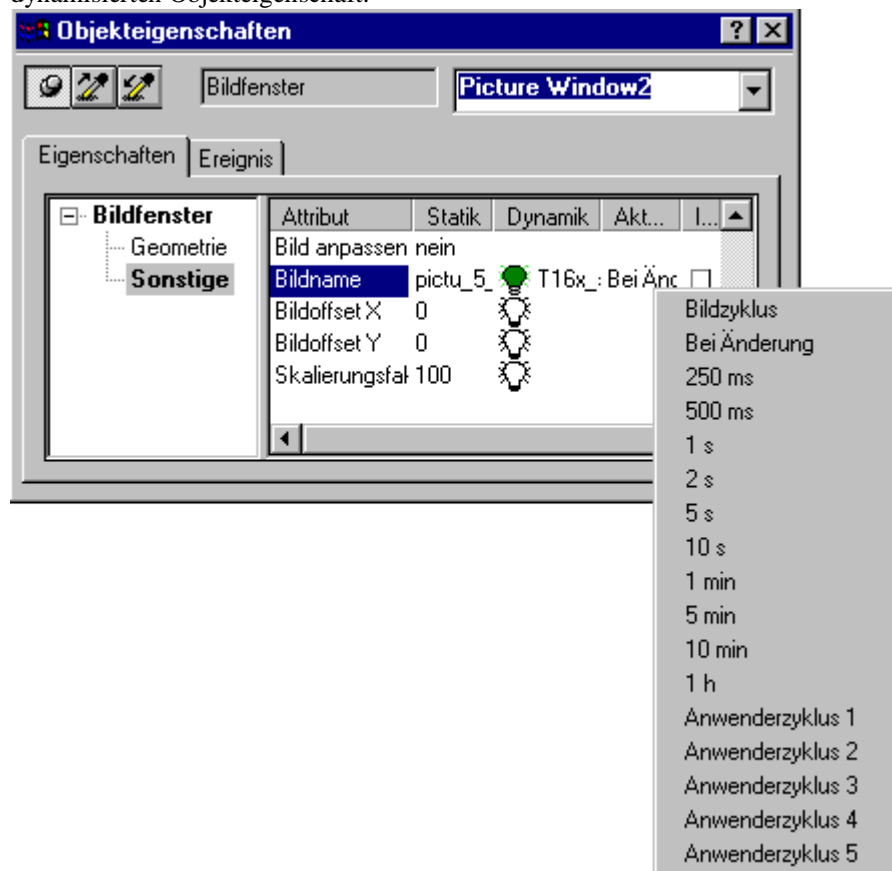
Dynamic Wizard

Diese Seite erscheint bei Anwahl des Punktes *Eigenschaft dynamisieren* auf der Registerkarte *Standard Dynamiken* im *Dynamic-Wizard* mit  **D**.



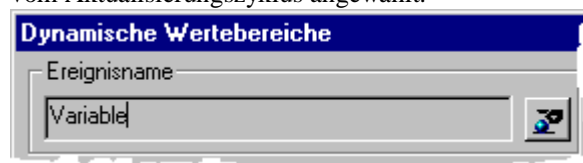
Variablenanbindung bei Objekt-Eigenschaft

Dieses Menü erscheint bei Anwahl der Spalte *Aktual.* mit  bei einer mit einer Variable dynamisierten Objekteigenschaft.



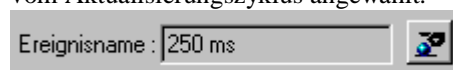
Dynamik-Dialog

Bei *Dynamik-Dialogen* wird durch Anwahl des Trigger-Button der Dialog zur Änderung vom Aktualisierungszyklus angewählt.



C-Aktion für Eigenschaft

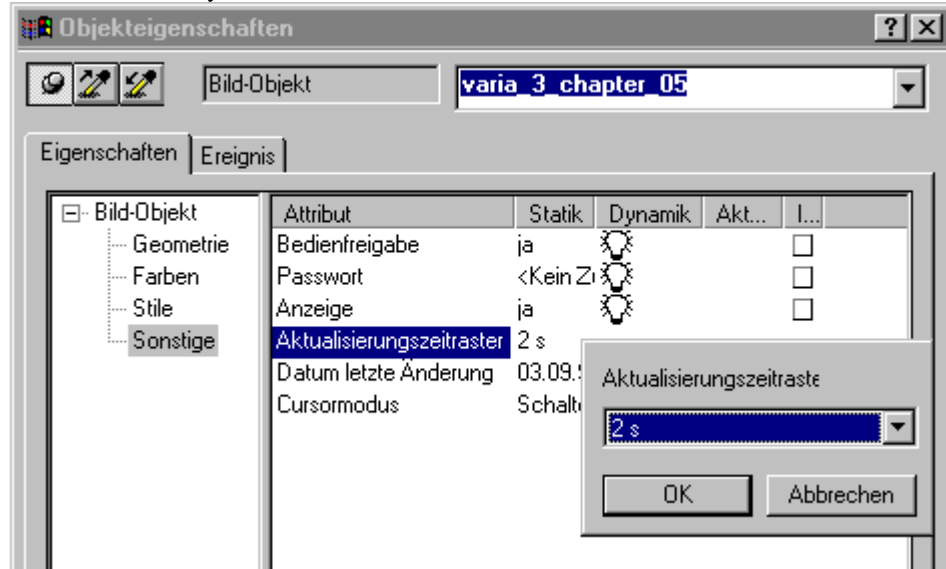
Bei *C-Aktion* wird im Editor durch Anwahl des Trigger-Button der Dialog zur Änderung vom Aktualisierungszyklus angewählt.



Die standardmäßig eingestellten Aktualisierungszyklen können folgendermaßen verändert werden:

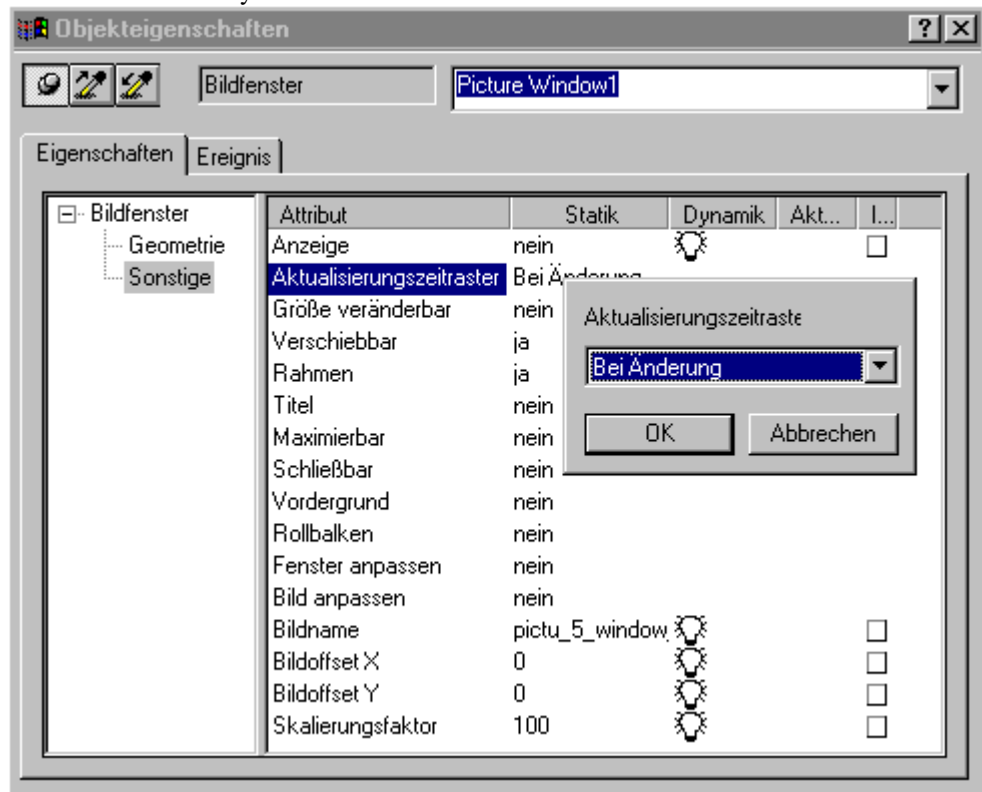
Bildzyklus

Ändern vom Bildzyklus



Fensterzyklus

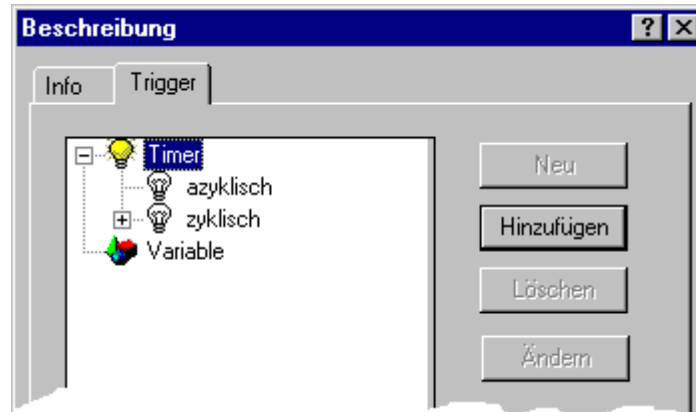
Ändern vom Fensterzyklus



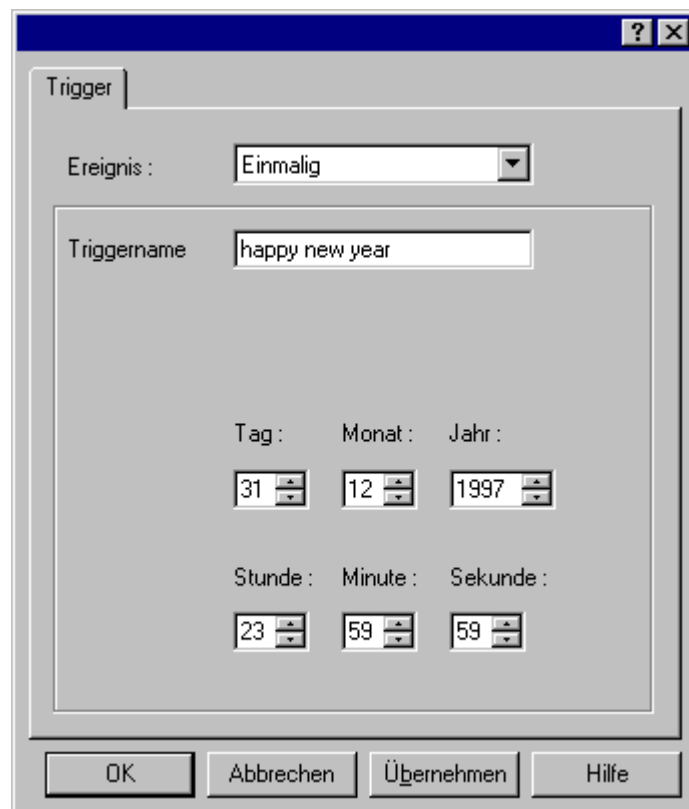
3.3.1.5 Ausführung von Hintergrundscripten (Global Script)

Die Ausführung von Hintergrundscripten (*Global Script*) ist je nach Projektierung von verschiedenen Größen abhängig:

- Zeittrigger (zyklische Ausführung, Ausnahme: azyklisch = einmalig)
 - Zeitzyklus
 - Zeitpunkt
- Ereignistrigger



oder nur einmal

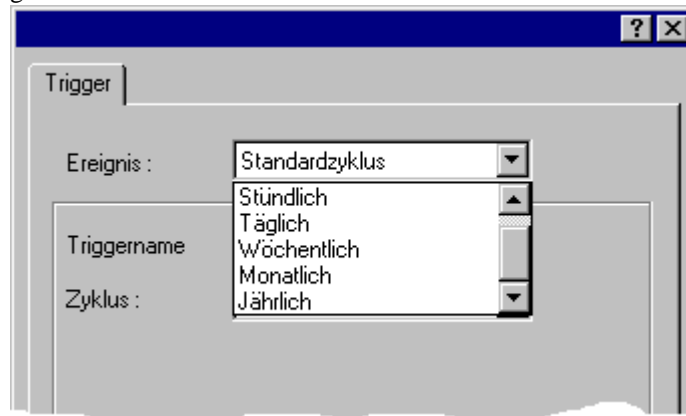


Zeitzyklus

Der projektierte Zeitfaktor der globalen Aktion bestimmt, **wann** die definierte Aktionsfolge bearbeitet werden muß. Neben dem bereits beschriebenen Standardzyklus und den zugehörigen Zeiteinstellungen von 250 ms bis 1 h (bzw. Anwenderzyklus1 bis 5), können auch die Zeit-Trigger:

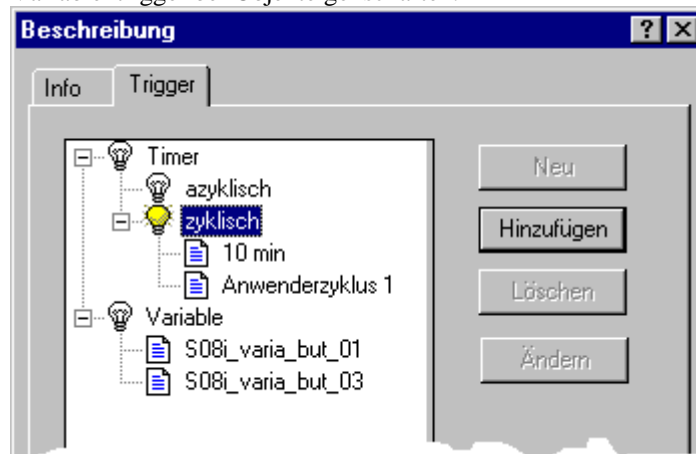
- Stündlich (Minute und Sekunde)
- Täglich (Stunde, Minute, Sekunde)
- Wöchentlich (Wochentag, Stunde, Minute, Sekunde)
- Monatlich (Tag, Stunde, Minute, Sekunde)
- Jährlich (Monat, Tag, Stunde, Minute, Sekunde)

gewählt werden.

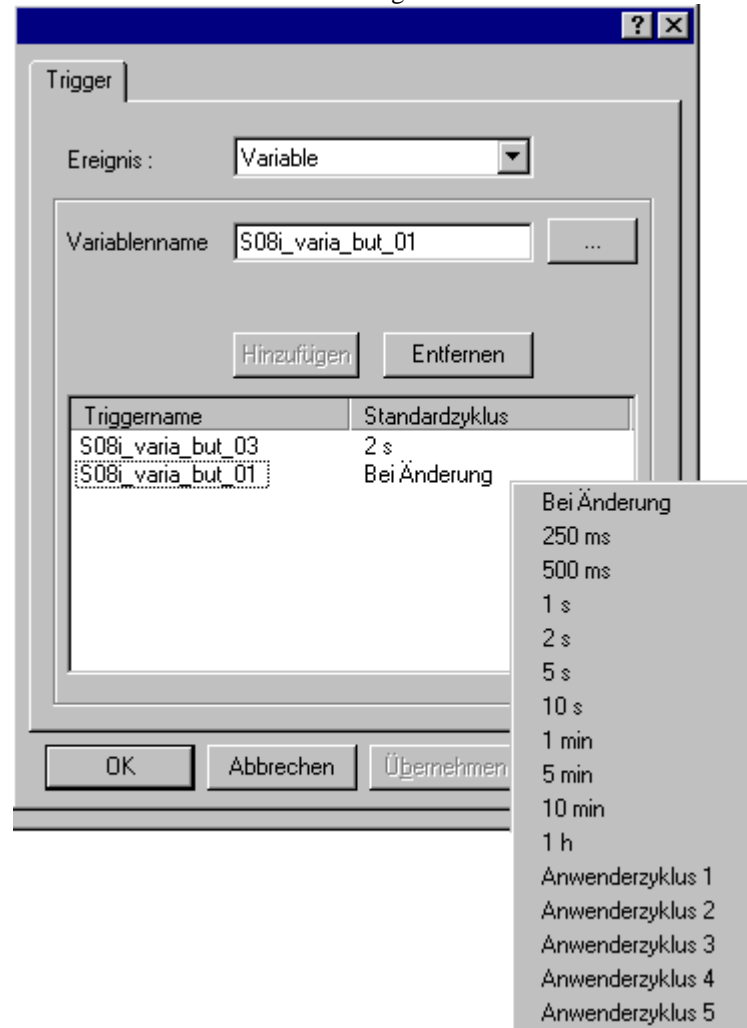


Variablentrigger

Wird die Aktion abhängig von einer oder mehreren Variablen aktiviert, so muß der Ereignistrigger als Variablentrigger gesetzt werden. Dies erfolgt analog dem Variablentrigger bei Objekteigenschaften.



Standardmäßig wird als Zeitrahmen der Zyklus von 2 Sekunden gesetzt. Der Projektteur kann anstatt des Default-Wertes folgende Zeitfaktoren einstellen:



Zu Beginn und am Ende des eingestellten Zeitrahmens wird der Wert der ausgewählten Variablen ermittelt. Hat sich der Wert mindestens einer Variablen geändert, so wird der Trigger für die globale Aktion ausgelöst.

Beachten Sie die hohe Systembelastung bei Triggerung auf Variable *bei Änderung*. Diese Einstellung ist nicht immer sinnvoll. Es gelten die gleichen Hinweise wie bei der Objekt Aktualisierung.

Alle Aktionen, die Sie als globale Aktionen definieren, werden **nicht objektgebunden**, d.h. nur abhängig von den eingestellten Zeitzyklen oder Ereignistriggern, geprüft und aktiviert. Setzen Sie daher die globalen Aktionen gezielt ein und vermeiden Sie unnötige Aktionsschritte, um das System nicht zu stark zu belasten. Verwenden Sie nicht zu viele und zu viele kleine Zeitzyklen für die Ausführung Ihrer Aktionen.

3.3.2 WinCC Dynamisierung

Definition

Unter Dynamisierung verstehen wir das Ändern von Zuständen (z.B. Position, Farbe, Schrift, u.a.m.) und das Reagieren auf Ereignisse (z.B. Mausklick, Tastaturbedienung, Wertänderung, u.a.m.) während der Laufzeit (Runtime).

Jedes Element im Grafikfenster wird als eigenständiges Objekt betrachtet. Das Grafikfenster selbst ist ebenfalls ein Objekt vom Typ *Bild-Objekt*.

Im WinCC *Grafik System* besitzt jedes Objekt *Eigenschaften* und *Ereignisse*. Mit wenigen Ausnahmen können diese dynamisiert werden. Diese Ausnahmen betreffen im wesentlichen *Eigenschaften* und *Ereignisse*, die im Runtime keine Auswirkung haben. Sie besitzen kein Symbol für die Dynamisierbarkeit.

3.3.2.1 Dynamisieren der Eigenschaften

Die Eigenschaften eines Objekts (Position, Farbe, Schrift, u.a.m.) können statisch gesetzt und im Runtime dynamisch verändert werden.

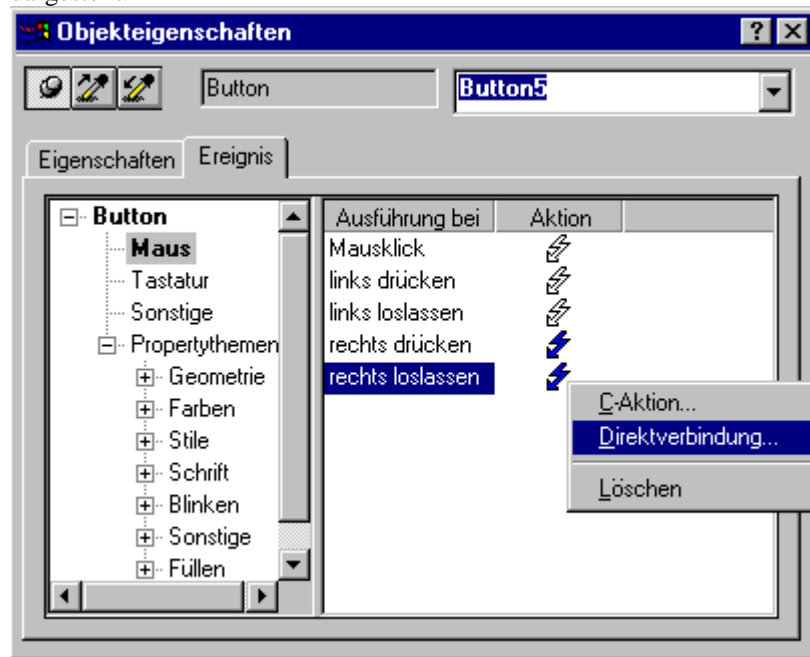
Alle Eigenschaften mit einer Glühbirne in der Spalte *Dynamik* sind dynamisierbar. Ist eine Eigenschaft dynamisiert, wird abhängig von der Dynamisierungsart ein farbiges Symbol anstelle der weißen Glühbirne angezeigt. Themen (z.B. Geometrie), die dynamisiert sind, werden Fett dargestellt.



3.3.2.2 Dynamisieren der Ereignisse

Die Ereignisse eines Objekts (z.B. Mausklick, Tastaturbedienung, Wertänderung, u.a.m.) können im Runtime abgefragt und dynamisch ausgewertet werden.

Alle Ereignisse mit einem Pfeilsymbol in der Spalte *Aktion* sind dynamisierbar. Ist ein Ereignis dynamisiert wird abhängig von der Dynamisierungsart ein farbiger Pfeil anstelle des weißen Pfeils angezeigt. Themen (z.B. Sonstige), die dynamisiert sind, werden Fett dargestellt.



3.3.2.3 Dynamisierungsarten für Objekte

Die Objekte eines Anlagenbildes können auf unterschiedliche Art und Weise dynamisiert werden. Die einzelnen Standarddialoge für die Dynamisierung sind für unterschiedliche Zielgebiete ausgerichtet und führen zum Teil auch zu unterschiedlichen Ergebnissen.

Übersicht








Dynamisierungsart	A	B	Vorteil	Nachteil
<i>Dynamic Wizard</i>	x	x	geführter Standardweg bei der Projektierung	nur für bestimmte Dynamisierungen. Erzeugt immer eine <i>C-Aktion</i> !
<i>Direktverbindung</i>		x	Die schnellste Dynamisierung im Bild, höchste Performance im Runtime	beschränkt auf eine Verbindung und nur innerhalb eines Bildes einsetzbar.
Variablenanbindung	x		Einfach zu projektieren	eingeschränkte Dynamisierungsmöglichkeiten
Dynamikdialog	x		Schnell und übersichtlich; für Wertebereiche oder mehrere Alternativen, hohe Performance im	nicht für alle Dynamisierungen einsetzbar

Dynamisierungsart	A	B	Vorteil	Nachteil
			Runtime	
C-Aktion	x	x	Fast unbegrenzte Möglichkeiten der Dynamisierung durch die leistungsfähige Scriptsprache (Ansi-C)	Fehlermöglichkeiten durch falsche C-Befehle geringere Performance gegenüber anderen Dynamisierungsarten, daher immer überprüfen, ob das Ziel nicht durch eine andere Dynamisierungsart erreicht werden kann.

Legende:

- A Dynamisierung der Objekteigenschaft
B Dynamisierung vom Objekt ereignis

Aufruf der Dialoge für die Dynamisierung

Dialog	Aufruf
Konfigurationsdialog	Nicht alle Objekte haben einen solchen Dialog. Automatisch beim Erzeugen dieser Objekte. <i>Objekt im Bild selektieren → SHIFT-Taste drücken und gedrückt halten → .</i> <i>Objekt im Bild selektieren →  das Kontextmenü öffnen → Konfigurationsdialog</i>
Dynamic Wizard	<i>Objekt im Bild selektieren → Eigenschaft oder Ereignis wählen → Wizard wählen und mit  starten . Der Dynamic Wizard muß über Ansicht → Symbolleisten... angewählt sein.</i>
Direktverbindung	<i>Objekt im Bild selektieren → Objekteigenschaften anzeigen → Register Ereignis anwählen → in der Spalte Aktion mit  das Kontextmenü öffnen → Direktverbindung wählen.</i>
Variablenanbindung	<i>Objekt im Bild selektieren → Objekteigenschaften anzeigen → Register Eigenschaften anwählen → in der Spalte Dynamik mit  das Kontextmenü öffnen → Variable wählen → im Dialog die entsprechende Variable auswählen und übernehmen.</i>
Dynamikdialog	<i>Objekt im Bild selektieren → Objekteigenschaften anzeigen → Register Eigenschaften anwählen → in der Spalte Dynamik mit  das Kontextmenü öffnen → Dynamik-Dialog. wählen → im Dialog die entsprechende Dynamik projektieren und übernehmen</i>
C-Aktion	<i>Objekt im Bild selektieren → Objekteigenschaften anzeigen → Register Eigenschaften oder Ereignis anwählen → in der Spalte Dynamik oder Aktion mit  das Kontextmenü öffnen → C-Aktion wählen → die entsprechende C-Aktion projektieren und übersetzen</i>

Ergebnisse und Darstellung

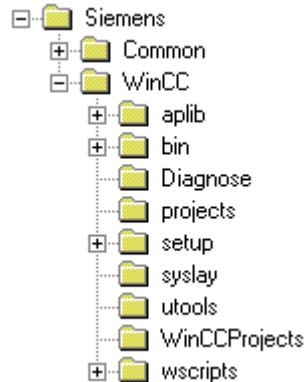
Dialog	Ergebnis	Darstellung
<i>Dynamic Wizard</i>	Es wird immer eine <i>C-Aktion</i> erzeugt.	<i>Grüner Pfeil</i>
<i>Direktverbindung</i>		<i>Blauer Pfeil</i>
<i>Variablenanbindung</i>		<i>Grüne Glühbirne</i>
<i>Dynamikdialog</i>	automatisch generierte <i>C-Aktion</i> (InProc), diese <i>C-Aktion</i> kann nachträglich erweitert werden, dabei geht aber der Performancevorteil verloren	<i>Roter Pfeil</i> bei Änderung in <i>C-Aktion</i> Wechsel zu <i>Grüner Pfeil</i>
<i>C-Aktion</i>	projektiertes C-Script	<i>Grüner Pfeil</i> <i>Gelber Pfeil</i> - die Aktion muß noch übersetzt werden

3.3.3 WinCC Systemumgebung

WinCC wird unter dem Standard-Installationspfad `C:\Siemens\WinCC\` angelegt. Während der Installation kann dieser Standardpfad geändert werden.

3.3.3.1 Verzeichnisstruktur WinCC System

Die Verzeichnisstruktur ohne Optionen und ohne Beispiele ist folgendermaßen aufgebaut.



Dateien im WinCC Standardverzeichnis

Im Standard-WinCC-Pfad sind für den Projektteur bzw. Inbetriebsetzer folgende Verzeichnisse und Dateien wichtig:

Verzeichnis	Dateiname, Extension	Bemerkung
Diagnose	License.log	aktuelle Logbucheinträge hinsichtlich der Lizenzprüfungen bzw. -verletzungen
	License.bak	die Logbuchdatei der Lizenzinformationen des letzten Startvorgangs
	WinCC_- Op_01.log	Operatormeldungen, die zur Laufzeit von WinCC erzeugt werden.
	WinCC_- Sstart_01.log	Systemmeldungen, die beim Anlauf von WinCC erzeugt werden. Eine wichtige Datei bei der Fehlersuche . Die Datei beinhaltet Meldungen über fehlende Variablen, fehlerhaft ausgeführte Scripte.
	WinCC_- Sys_01.log	Systemmeldungen, die zur Laufzeit von WinCC erzeugt werden. Eine wichtige Datei bei der Fehlersuche . Die Datei beinhaltet Meldungen über fehlende Variablen, fehlerhaft ausgeführte Scripte.
	S7chn01.log	Systemmeldung des verwendeten Kanales (hier S7)
aplib	Bibliotheks- pfad	die header-Dateien, alle Standard-Funktionen und alle Internen Funktionen, sind in Unterverzeichnissen abgelegt.

Dateien im WinCC Standardverzeichnis

Im Standard-WinCC-Pfad sind die projektübergreifenden Funktionen und Symbole in folgenden Verzeichnissen abgelegt:

Verzeichnis	Unterverzeichnis, Dateiname	Bemerkung
aplib	library.pxl	Symbole der Standard-Bibliothek von WinCC
	Report, Wincc, Windows	Verzeichnisse für Standard-Funktionen, diese können jederzeit angepaßt werden.
	Allocate, C_bib, Graphics, Tag	Verzeichnisse für Interne-Funktionen, diese können nicht angepaßt werden.
syslay		alle Drucklayouts, die beim Anlegen eines Projektes von WinCC automatisch in den Projektpfad ins Verzeichnis <i>prt</i> kopiert werden.
wscripts	Dynwiz.cwd	<i>Dynamic Wizard</i> des <i>Graphics Designer</i> . Es können jederzeit eigene Scripte erstellt werden. Diese Scripte erhalten jeweils die Endung .wnf
	wscripts.deu	dieser Pfad enthält die Scriptdateien für <i>deutsch</i> . Dieser Pfad ist abhängig von der installierten Sprache.
	Wscripts.enu	dieser Pfad enthält die Scriptdateien für <i>englisch</i> . Da englisch die Defaultsprache ist, wird dieser Pfad immer angelegt.
	Wscripts.fra	dieser Pfad enthält die Scriptdateien für <i>französisch</i> . Dieser Pfad ist abhängig von der installierten Sprache.

Dateien im WinCC Standardverzeichnis

Folgende Anwendungsprogramme werden bei der Installation von WinCC in folgenden Verzeichnissen abgelegt:

Verzeichnis\ Datei	Bemerkung
\sqlany\isql.exe	Interaktives Programm, um die Daten in der Datenbank eines WinCC-Projektes einsehen zu können.
\bin\Wunload.exe	<p>Assistent (Wizard), um die Online-Tabellen in der Datenbank des WinCC-Projektes zu leeren, z.B. Entfernen der abgelegten Meldungen und Meßwertdaten.</p> <p>Der Assistent stellt die Laufzeittabellen automatisch für die Entladung ein; weitere Tabellen können jedoch vom Anwender jederzeit aus der Liste hinzugefügt bzw. entfernt werden.</p> <p>Dieses Tool muß offline zu einem WinCC-Projekt eingesetzt werden. Im Runtime-Modus kann dieses Tool nicht genutzt werden. Das Auslagern von Meldungen und Meßwerten zur Laufzeit (Runtime-Modus) kann mittels des Optionspaketes <i>STORAGE</i> durchgeführt werden.</p>

Verzeichnis\ Datei	Bemerkung
\bin\Wrebuild.exe	Assistent (Wizard) für die Rekonstruktion der Datenbank, der nicht im Runtime-Modus eingesetzt werden kann!
\SmartTools\CC_GrafikTools\metaVw.exe	Viewer für Grafikdateien (z.B. Druckaufträge, exportierte Symbole) im EMF-Format (extended meta file).
\SmartTools\CC_GrafikTools\wmfdcode.exe	Viewer für Grafikdateien im WMF-Format (windows meta file).
\SmartTools\CC_OCX_REG\ocxreg.exe	Zur Registrierung bzw. zur Aufhebung der Registrierung von weiteren OLE Control Komponenten (OCX).
\SmartTools\CC_OCX_REG\Regsvr32.exe	wird von ocxreg.exe aufgerufen.

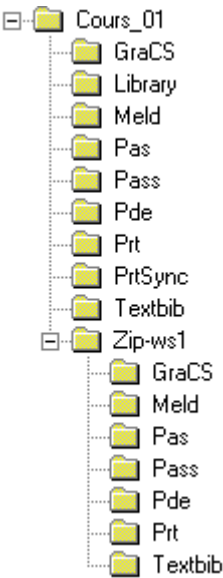
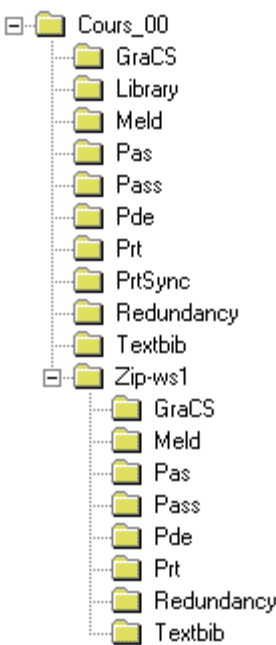
3.3.4 WinCC Projektumgebung

Hinweis:

Legen Sie für WinCC Projekte ein spezielles Projektverzeichnis an, z.B. WinCC_Projekte. Es wird eine klare Trennung zwischen dem WinCC System und den projektierten Daten erzielt. Auf diese Weise vereinfacht sich die Datensicherung. Die Gefahr des Datenverlustes (durch Bedienfehler) bei einer Deinstallation von WinCC wird weitestgehend umgangen.

3.3.4.1 WinCC Projekt - Verzeichnisstruktur

Ein Projekt unter WinCC besteht aus einer Verzeichnisstruktur mit entsprechendem Inhalt. Nach dem Anlegen eines neuen Projektes im *Control Center* (über den Menüpunkt *Datei* → *Neu*) wird eine Verzeichnisstruktur folgendermaßen aufgebaut:

WinCC Standard	WinCC mit Optionen
 <pre> graph TD C1[Cours_01] --> G1[GraCS] C1 --> L1[Library] C1 --> M1[Meld] C1 --> P1[Pas] C1 --> Pa1[Pass] C1 --> Pde1[Pde] C1 --> Pr1[Prt] C1 --> Prs1[PrtSync] C1 --> T1[Textbib] C1 --> Z1[Zip-ws1] Z1 --> G2[GraCS] Z1 --> M2[Meld] Z1 --> P2[Pas] Z1 --> Pa2[Pass] Z1 --> Pde2[Pde] Z1 --> Pr2[Prt] Z1 --> T2[Textbib] </pre>	 <pre> graph TD C2[Cours_00] --> G3[GraCS] C2 --> L2[Library] C2 --> M2[Meld] C2 --> P2[Pas] C2 --> Pa2[Pass] C2 --> Pde2[Pde] C2 --> Pr2[Prt] C2 --> Prs2[PrtSync] C2 --> R2[Redundancy] C2 --> T2[Textbib] C2 --> Z2[Zip-ws1] Z2 --> G4[GraCS] Z2 --> M4[Meld] Z2 --> P4[Pas] Z2 --> Pa4[Pass] Z2 --> Pde4[Pde] Z2 --> Pr4[Prt] Z2 --> R4[Redundancy] Z2 --> T4[Textbib] </pre>

Inhalt der Projektverzeichnisse

Verzeichnis	Extension	Bemerkung
Projektpfad	.db	die Datenbank mit den Projektierungsdaten
	rt.db	die Datenbank mit den Laufzeitdaten (Runtime) Meßwerte, Meldungen
	.mcp (master control program)	Hauptdatei des WinCC Projektes. Mit dieser Datei wird das Projekt geöffnet.
	.pin	Projekt.pin
GraCS	.pdl (picture design language)	die projektierten Bilder
	.sav	Sicherungsdateien der Bilddateien mit dem letzten Projektierungszustand
	.bmp (bitmap), .wmf (windows meta file), .emf (extended meta file)	Bilddateien
	.act (action)	exportierte C-Aktionen
	.pdd	<i>Default.pdd</i> Einstellungsparameter für den Grafikeditor (Default-Einstellungen der einzelnen Objekte in der Objektpalette)
Library	.h (header)	<i>Ap_pbib.h</i> (Projekt-Funktionsdeklarationen)
	.pxl	<i>Library.pxl</i> (Projekt-Symbolbibliothek)
	.fct	Projektfunktionen
	.dll (dynamic link library)	eigene Funktionsbibliotheken, die mit einer C Entwicklungsumgebung erstellt wurden.
Meld		
Pas	.pas (Aktionsdefinition)	Projekt-Aktionen, die als Hintergrundaktionen abhängig vom eingestellten Trigger ablaufen
Pass		
Pde		
Prt	.rpl (report picture language) .rpl (Zeilenlayout)	Seitenlayouts für Druckaufträge Die vordefinierten WinCC-Standard-Layouts beginnen jeweils mit @. Alle Systemgrößen (auch Variable) sind mit diesem Präfix gekennzeichnet.
Rechnername z.B. Zip-ws1	\GraCS\GraCS. ini	Initialisierungsdatei für den Grafikeditor

Optional: Dateien die während der Projektierung entstehen können

Verzeichnis	Extension	Bemerkung
zum Teil frei bestimmbar	.ini	Initialisierungsdatei für den Simulator mit Informationen für den Aufruf.
	.sim	interne Variablen mit Einstellungen für die Simulation
	.csv	exportierte Texte aus der Textbibliothek
	.txt	exportierte Meldungen aus dem Meldesystem (<i>Alarm Logging</i>)
	.emf	Druckaufträge, die Ihre Druckergebnisse in eine Datei schreiben
	.log	Logdateien
	.xls .doc .wri	Dateien, die mit anderen Anwendungen erstellt wurden, aber im WinCC-Projekt genutzt werden

3.3.5 Automatischer Projektanlauf von WinCC

Anforderung

Das HMI-System (WinCC) auf der Anlage soll nach dem Einschalten des Windows-Systems automatisch starten. Der Operator an der HMI-Station benötigt keine Kenntnisse für die Bedienung des Windows-Systems (z.B. Aufruf von WinCC unter Windows 95 oder unter Windows NT).

Lösung

WinCC wird in der Anlaufroutine des PC's automatisch beim Hochlauf gestartet. Diese Einstellung wird im Ordner *Autostart* von Windows vorgenommen.



Verknüpfung anlegen

Schritt	Vorgehen bei NT4.0:
1	Im Windows Explorer in das Verzeichnis <i>WinNT\Profiles\All Users\Startmenü\Programme\Autostart</i> wechseln. WinNT ist das Verzeichnis in dem Windows NT installiert wurde
2	Im Ordner eine neue Verknüpfung mit der <i>UR → Neu → Verknüpfung anlegen</i>
3	Die Verbindung zum Programm <i>mcp.exe</i> (<u>M</u> aster <u>C</u> ontrol <u>P</u> rogram) im <i>\bin</i> Verzeichnis von WinCC herstellen.
4	Der Verbindung einen Namen geben.

Dadurch wird das WinCC *Control Center* automatisch gestartet. WinCC selbst startet automatisch mit dem Projekt, das zuletzt bearbeitet wurde oder aktiviert war. Um eine Anlage im Runtime-Modus zu starten, muß also das Projekt im aktiven Zustand verlassen werden.

Hinweis:

Ist die Tastenkombination *CTRL + SHIFT* nicht gesperrt und wird sie beim Anlauf von WinCC gedrückt, startet WinCC im Projektiermodus, auch wenn das Projekt im aktiven Modus beendet wurde.

Der Operator sieht nun sein bekanntes Startbild der Anlage. Damit der Operator nicht ungewollt bzw. absichtlich in die Projektierung (die im Hintergrund läuft) umschalten kann bzw. Windows Applikationen nutzen kann, die er nicht bedienen soll oder darf, müssen entsprechende Vorkehrungen getroffen werden. Der Operator darf auch nicht in das Datenbank Laufzeitfenster gelangen, da er über diesen Weg die WinCC Datenbankverbindung schließen kann.

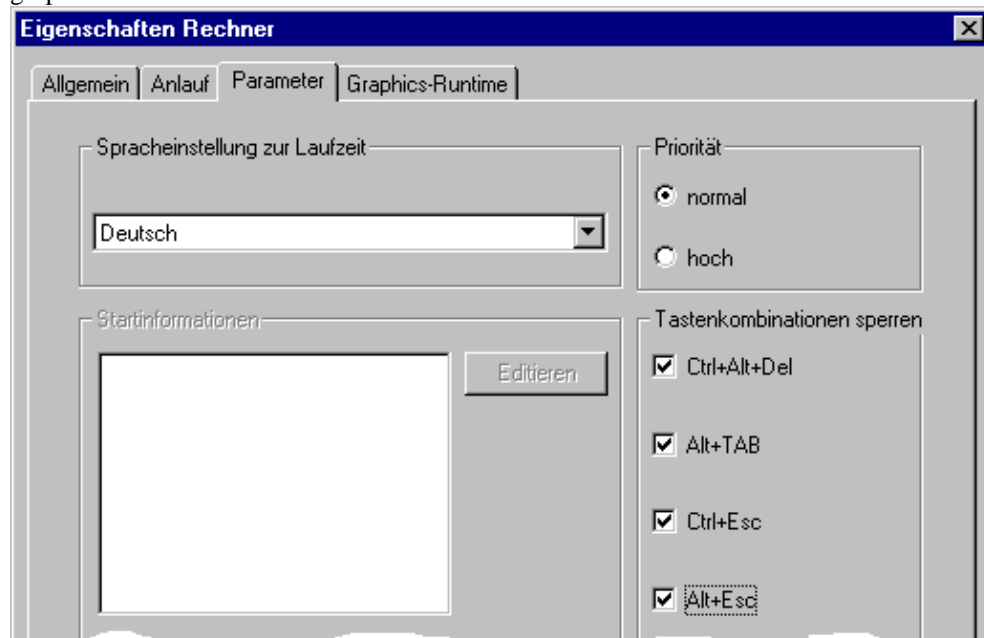
Keine Anwahl von:

Der Operator sollte keine Möglichkeit haben vom WinCC Runtime zu wechseln in:

- das *Control Center* von WinCC (Projektierungsumgebung),
- das Laufzeitfenster der *SQL-Datenbank* von WinCC (Sybase SQL Anywhere), da über diesen Weg ein Anwender die WinCC Datenbankverbindung schließen kann. WinCC kann dann nicht mehr weiterarbeiten,
- die *Taskleiste* von Windows, da über diesen Weg **alle** installierten Programme gestartet werden können,
- das aktuelle Taskfenster, da die Applikation geschlossen werden kann.

Notwendige Einstellungen des Rechners

Damit der Operator diese Möglichkeiten nicht hat, müssen folgende Tastenkombinationen gesperrt werden.

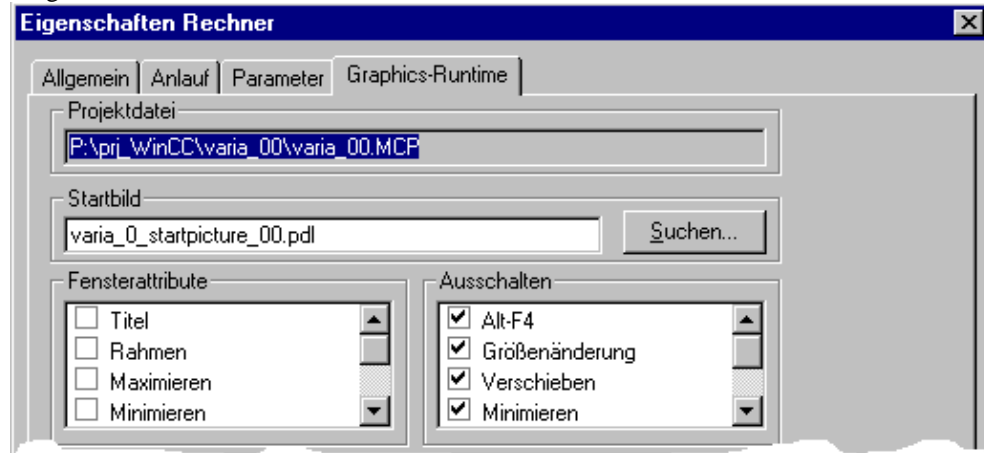


Das Sperren erfolgt im *Control Center* im Dialog *Eigenschaften Rechner*.

Die genaue Bedeutung der einzelnen Tastenkombinationen kann aus der Hilfe zu WinCC bzw. der Hilfe zum jeweiligen Betriebssystem entnommen werden.

Notwendige Einstellungen des Runtime:

Über die Standard-Windows-Tasten könnte das Anlagenbild geschlossen werden; d.h. über diesen Weg könnte das Verlassen von WinCC auch erreicht werden. Um dies zu verhindern, müssen folgende Einstellungen für die Eigenschaften des Anlagenbildes vorgenommen werden:



Das Sperren erfolgt im *Control Center* bei *Eigenschaften Rechner*.

Die genaue Bedeutung der einzelnen Tastenkombinationen kann aus der Hilfe zu WinCC bzw. der Hilfe zum jeweiligen Betriebssystem entnommen werden.

- wenn *Größenänderung* und *Minimieren* nicht ausgeschaltet sind, wird die Oberfläche des Betriebssystems wieder zugänglich.
- *Schließen* (im Bild nicht dargestellt) muß ebenfalls ausgeschaltet werden. Andernfalls kann das Runtime beendet werden und der Anwender gelangt in das Configuration System.

Hinweis:

Werden die genannten Tasten alle oder teilweise gesperrt, muß dem Projektteur bzw. dem Servicepersonal der Zugang zur Projektierung mittels einer eigens dafür projektierten Taste ermöglicht werden. Dies gilt auch für das ordentliche Herunterfahren der Anlage.

Diese Funktionen dürfen nicht frei bedienbar sein. Es ist zusätzlich ein **Zugriffsschutz**, z.B. Service Personal, hinter der Eigenschaft *Paßwort* des Buttons zu hinterlegen.

3.3.6 Koordiniertes Beenden von WinCC

Beenden durch Bedienung

Eine WinCC-Station darf niemals ohne Shutdown des Betriebssystems ausgeschaltet werden. Ein *NOTAUS-Schalter* ist für das HMI-System nicht geeignet. Deshalb muß für den Operator eine entsprechende Bedientaste projektiert werden, die Ihm erlaubt, ohne zusätzliches Wissen das System korrekt zu beenden.

Stromausfall

Um Datenverluste durch Stromschwankungen bzw. Stromausfall möglichst zu vermeiden, sollte ein detailliertes **Datensicherungskonzept** für das HMI-System ausgearbeitet und angewendet werden.

Prinzipiell sollte für die WinCC-Station auf jeden Fall eine USV (Unterbrechungsfreie Stromversorgung) eingeplant werden. Dies kann der Anschluß an die betriebsinterne USV sein oder der Anschluß einer eigenen USV an den Server von WinCC. Dies gilt sowohl für Einplatz- als auch für Mehrplatzsysteme unabhängig vom eingesetzten Betriebssystem (Windows 95 oder Windows NT).

Die eingesetzte USV muß zusätzlich eine spezielle Shutdown-Software für Windows 95 bzw. Windows NT beinhalten, damit bei einem Stromausfall und dem Abschalten nach einer bestimmten Zeitspanne das Betriebssystem und alle aktiven Applikationen ohne Datenverlust automatisch heruntergefahren werden; z.B.: APC USV 600 mit Power-Shutdown Software für Windows 95 und Windows NT.

Hinweise zur Installation einer USV

Für den Anschluß einer USV mit entsprechender Prüfsoftware muß eine serielle Schnittstelle zur Verfügung stehen. Falls an der WinCC Station keine serielle Schnittstelle frei ist, z.B. belegt durch Drucker- oder PLC-Anschluß, muß eine zusätzliche Schnittstellenkarte eingesetzt werden.

Mehrfach genutzte serielle Schnittstellen (z.B. über Switches) werden von den meisten USV-Systemen nicht unterstützt und sind auch nicht sinnvoll, da eine ständige Überwachung des Systems nötig ist.

Im Betriebssystem wird ein entsprechender Überwachungsdienst installiert. Dieser Überwachungsdienst muß noch mit Shutdown Parametern versorgt werden, um ein koordiniertes Beenden des Systems zu garantieren. Der Shutdown Prozeß für Applikationssoftware muß auf jeden Fall aktiviert werden, damit WinCC bei einem Shutdown ohne Datenverlust heruntergefahren wird. Abhängig von den aktiven Applikationen muß die *Sicherungszeit* vor dem Shutdown ausreichend groß gewählt werden.

Die Software für die USV-Systeme bieten meistens auch einen zeitlich gesteuerten *Shutdown* z.B. für das Wochenende oder für die Nacht. Über diesen Weg kann ein **gezieltes Beenden des WinCC-Systemes ohne Bedienung** erreicht werden.

3.3.7 Datensicherung

Wann soll gesichert werden?

Ein WinCC-Projekt muß aus folgenden Gründen wiederholt gesichert werden:

- Datensicherung während der Projektierungsphase.
- Sicherung vor dem Export bzw. Import von Daten (z.B. beim Import von Variablen, von mehrsprachigen Bildtexten, von Meldungstexten sowie von mehrsprachigen Meldungstexten).
- Sicherung vor der Neubildung bzw. dem Entladen der WinCC-Datenbank.
- Sicherung vor der Bearbeitung der Datenbank mit Tools wie z.B. interaktiver SQL-Zugriff.
- Sicherung der Projektierungsdaten für die Installation auf der Zielanlage beim Endkunden.
- Übernahme von Daten für ein ähnlich aufgebautes Projekt.

Welche Medien sind geeignet?

Medium	Vorteil	Nachteil
Disketten	fast überall lesbar	zu geringe Kapazität (auch in gepackter Form)
ZIP-Disketten	preiswert, ausreichende Kapazität; direkter und schneller Zugriff über Windows möglich; einfach installierbar; mobil, für den Einsatz auf der Anlage	
Streamer (z.B. im Netzwerk)	automatische Sicherung möglich (täglich), sehr große Kapazität	meist nur in Büroumgebung vorhanden, kein direkter Zugriff auf die Daten da spezielles Format beim Speichern
Festplatte auf einem anderen PC (z.B. Laplink)	kein Medien-Handling, Daten direkt verwendbar	langsam, bei großen Datenmengen eher ungeeignet
MOD	hohe Datensicherheit, wiederverwendbar, Sicherung von Meldungen und Meßwerten im Runtime-Modus möglich.	für das Schreiben und Lesen ist ein spezielles Laufwerk notwendig
CD-ROM	hohe Kapazität, fast überall lesbar, für Langzeitarchivierung geeignet.	für das Beschreiben ist ein spezielles Laufwerk notwendig, nicht wiederverwendbares Medium

Projekt vor der Datensicherung bereinigen

Um das Projekt vor der Datensicherung für den Endkunden bzw. für die Übernahme von Projektdaten möglichst **bereinigt** zu sichern, können vorher folgende Daten gelöscht bzw. mittels Zusatzprogrammen bereinigt werden.

- Alle Sicherungsdateien im Verzeichnis `\GraCS*.sav` des Projektes.
- Wenn für die Dokumentation keine eigenen Layouts (Report Designer) erstellt wurden, können die Systemlayouts aus dem Verzeichnis `\Prt` gelöscht werden. Beim Anlegen eines neuen Projektes werden automatisch alle Systemlayouts aus dem Verzeichnis `\Siemens\WinCC\syslay` in das Projektverzeichnis kopiert.

Was muß gesichert werden?

Sollen nur die Daten eines WinCC-Projektes gesichert werden, müssen folgende Dateien und Verzeichnisse mit ihren Dateien gesichert werden.

aus dem Projektverzeichnis:

- die Dateien `*.mcp`, `*.pin`, `*.db`
- die Verzeichnisse `\GraCS` und `\Library`
- falls eigene Aktionen erstellt wurden, das Verzeichnis `\Pas`
- falls eigene Drucklayouts erstellt wurden, das Verzeichnis `\Prt`.

Wenn projektübergreifende Komponenten (Standardfunktionen, Objekte in der Projektbibliothek) erstellt wurden, müssen zusätzlich folgende Dateien aus

dem WinCC-Standardverzeichnis:

- die Dateien `\Siemens\WinCC\aplib*.fct`
- die Datei `\Siemens\WinCC\aplib\library.pxl`

gesichert werden. Diese Daten werden bei einer Neuinstallation von WinCC **nicht** generiert.

3.3.8 Kopieren eines gesicherten WinCC-Projektes auf eine neue Ziel-Maschine

Installation der Systemsoftware

Installation der WinCC-Software. Dies erfolgt bei Komplettsystemen über den automatisch aufgerufenen Konfigurationsdialog oder über die Installations-CD, die mit dem WinCC-Paket und den zugehörigen Lizenzdisketten geliefert werden.

Zusätze

Werden in Ihrem Projekt spezielle Zusatzpakete (z.B. Optionspakete oder Addons) oder spezielle Kommunikations-Schnittstellen oder Schnittstellen zu anderen Windows Programmen (z.B. WORD, EXCEL etc.) verwendet, so sind diese Pakete ebenfalls auf der Zielmaschine zu installieren. Die zugehörigen Autorisierungen für Optionspakete, Addons oder Kommunikations-Schnittstellen (Kanal-DLLs) sind auch auf der Zielmaschine zu installieren. Beachten Sie, daß **alle** nötigen Autorisierungen (für alle genutzten Kanal-DLLs) eingespielt werden müssen, um mit dem WinCC-Projekt arbeiten zu können.

Windows Software

Werden in den WinCC-Bildern OLE-Verbindungen zu anderen Windows Programmen, z.B. zu WORD, ClipArts oder EXCEL verwendet, so muß abhängig von der Art der OLE-Verbindung auf der Zielmaschine das zugehörige Softwarepaket ebenfalls installiert sein, d.h. in der Windows Registrierung eingetragen sein.

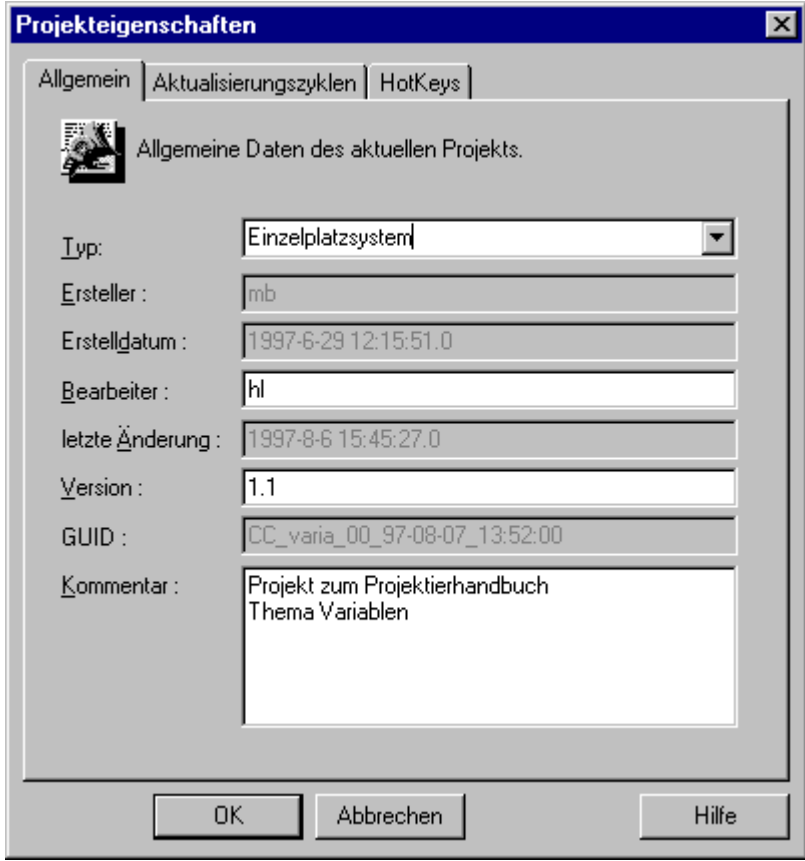
OXC, ActiveX

Werden weitere OCX-Komponenten (OLE Control, ActiveX) aus zugekauften Softwarepaketen eingesetzt, so müssen diese ebenfalls in der Windows Registrierung hinterlegt sein. Registrieren bzw. prüfen Sie die Registrierungseinträge dieser OCX-Komponenten z.B. mittels des auf der WinCC CD mitgelieferten Tools *SmartTools\CC_OCX_REG\ocxreg.exe*. Wird eine Registrierung zu OLE-Objekten bzw. OCX-Objekten beim Anlauf des WinCC-Projektes im Runtime-Modus (als auch in der Projektierung mittels des *Graphics Designer*) nicht gefunden, so wird dieses Objekt im Bild mit dem Hinweis *Unknown Object* eingeblendet.

Netzwerk

Wurde das Projekt für ein **Mehrplatzsystem** konfiguriert, muß vor dem Kopieren der WinCC Daten die gesamte Netzinstallation auf der WinCC Zielmaschine durchgeführt werden. Notieren Sie sich die nötigen Rechnernamen in der konfigurierten Rechnerlandschaft, da diese bei der Parametrierung des kopierten Projektes benötigt werden. Der Rechnername wird auch bei einem Einplatzsystem als Parameter benötigt, daher müssen Sie auf jeden Fall die Namensgebung auf der Zielmaschine kennen bzw. über die Windows Systemsteuerung ermitteln. Handelt es sich um ein Einplatzsystem ohne Netzanbindung, so wird der Windows Rechnername eingestellt sein.

Kopieren der Daten und Starten des Projektes

Schritt	Vorgehen beim der Datenübernahme
1	Anlegen eines Projektverzeichnisses (z.B. <i>WinCC_Projekte</i>)
2	Den gesamten gesicherten Pfad in dieses eingerichtete Verzeichnis als Unterverzeichnis kopieren (z.B.: <i>\WinCC_Projekte\Varia_00</i>). Der Name des WinCC-Projektverzeichnisses kann, falls gewünscht, geändert werden. Beim Umbenennen der Dateien im Projektverzeichnis (<i>varia_00.mcp</i> , <i>varia_00.db</i> , <i>varia_00.pin</i> , <i>varia_00.log</i>) ist darauf zu achten, daß alle Dateien den gleichen Namen (außer der Erweiterung) erhalten.
3	Das Projekt im WinCC <i>Control Center</i> öffnen.
4	Bei Bedarf die projektspezifischen Einstellungen ändern. Für die Änderung des Typs sind zusätzliche Anpassungen bei den <i>Rechnereigenschaften</i> notwendig. Diese Anpassungen sind in der WinCC Hilfe beschrieben. 
5	Den <i>Rechnernamen</i> überprüfen und gegebenenfalls im Dialog <i>Eigenschaften Rechner</i> auf der Registerkarte <i>Allgemein</i> ändern. Stimmt der Rechnername im WinCC-Projekt nicht mit dem Rechnernamen dieses Zielsystems überein, wird beim Aktivieren des Projektes, d.h. beim Anlauf im Runtime-Modus, eine Fehlermeldung angezeigt.
6	Wurden im Projekt eigene <i>Standard-Funktionen</i> (*.fct) verwendet, müssen die gesicherten Standard-Funktionen in den WinCC-Standardpfad <i>\Siemens\WinCC\aplib</i> kopiert und anschließend im Funktionsbaum bekannt gegeben werden. <ul style="list-style-type: none"> im geöffneten Projekt den Editor <i>Global Script</i> aufrufen

Schritt	Vorgehen beim der Datenübernahme
	<ul style="list-style-type: none">• die Deklarationsstruktur über den Menüpunkt <i>Optionen</i> → <i>Header neu generieren</i> neu erstellen. Nun sind die neuen Funktionen im Funktionsbaum sichtbar.
7	Das Projekt aktivieren, um den korrekten Anlauf des Projektes zu überprüfen.

3.3.9 Wiederverwendung - Übernahme von Projektteilen in ein neues bzw. bestehendes Projekt

Der Grund für die Übernahme

Ein WinCC-Projekt kann auf unterschiedliche Weise entstehen. Dabei sind die wichtigsten Gesichtspunkte die Wiederverwendbarkeit bereits bestehender Projektteile aus ähnlichen Projekten bzw. die Datenübernahme aus vorprojektierten Beispielprojekten.

Projektteam

Bei einem Projektierungsteam werden in dieser Hinsicht ähnliche Aufgaben zugrunde gelegt, da ein WinCC-Projekt letztendlich wieder zu einem Projekt verschmolzen werden muß.

Ein WinCC-Projekt besteht aus einzelnen Dateien (z.B. Bilder) **und** den Projektierungsdaten in der Datenbank (Meldesystem, Variablenhaushalt).

Daten in der Datenbank

Daten, die in der Projektierungsdatenbank hinterlegt sind, können **nicht** in zwei Einzelprojekten erstellt und anschließend zusammengeführt werden.

Aus diesem Grunde muß bei der Projektierung von Datenbankinformationen (z.B. Aufbau des Meldesystems) ein **Basisprojekt** erstellt werden, das für diese Art der Projektierung verwendet wird. Dieses Basisprojekt muß vor **jeder** Datenbankänderung (auch bei Zwischenschritten) **gesichert** werden. Geht bei der Änderung etwas schief kann auf den Stand vor der Änderung zurückgegriffen werden.

Hinweis:

Beachten Sie hierbei, daß jede Änderung in der Datenbank die Struktur und den Zugriff auf die Datenbank beeinflussen kann. Viele unnötige Änderungen (mit evtl. Löschvorgängen) können dazu führen, daß die WinCC Datenbank nicht mehr optimal zusammengestellt ist (Performanceverlust).

Einplatzsystem

Während an dem Basisprojekt der nächste Projektierungsschritt z.B. beim Meldesystem durchgeführt wird, sollte bei einem WinCC *Einzelplatzsystem* auf keinen Fall an anderer Stelle eine Datenbankänderung vorgenommen werden (z.B. Archivierung – *Tag Logging*).

Mehrplatzsystem

Wird dagegen auf einem WinCC *Mehrplatzsystem* an einem Projekt projektiert, so können gleichzeitig an unterschiedlichen Datenbankbereichen Projektierungen vorgenommen werden. Zum Beispiel kann ein Projektteur das Meldesystem und ein weiterer Projektteur das Archivierungssystem (Meßwerterfassung) bearbeiten.

Umwandlung Einzelplatz- Mehrplatzsystem

Jedes Projekt kann auf einem WinCC *Mehrplatz-Projektierungs-System* erstellt und auf der Ziel-Maschine wieder in ein Einzelplatzsystem umgewandelt werden. Bei einem vorgesehenen Wechsel der Konfiguration zwischen Projektierungs-Station(en) und Kundensystem dürfen keine spezifischen WinCC-Elemente genutzt werden, die auf ein Mehrplatzsystem ausgerichtet sind. Zum Beispiel sollten keine rechnerlokalen internen Variablen genutzt werden, wenn eigentlich ein Einzelplatzsystem beim Kunden konfiguriert wird.

Beim Anlegen eines neuen Projektes (entweder Einzelplatz- oder Mehrplatzsystem) muß bereits bekannt sein, ob auf einem bereits vorhandenen Basisprojekt mit vorprojektiertem Meldesystem bzw. Archivierungsdaten aufgesetzt werden soll. Diese in der Datenbank hinterlegten Daten sind nur durch Neuprojektierung oder wenn möglich durch Export - Import in andere Projekte übertragbar.

3.3.9.1 Übernahme von Bildern

Projektierte Bilder können jederzeit übernommen werden. Sie können die Bilddateien (*.pdl) entweder direkt über den Windows Explorer von dem Quellverzeichnis in das Zielverzeichnis des WinCC-Projektpfades \GraCS kopieren (geeignet bei mehreren Bilddateien).

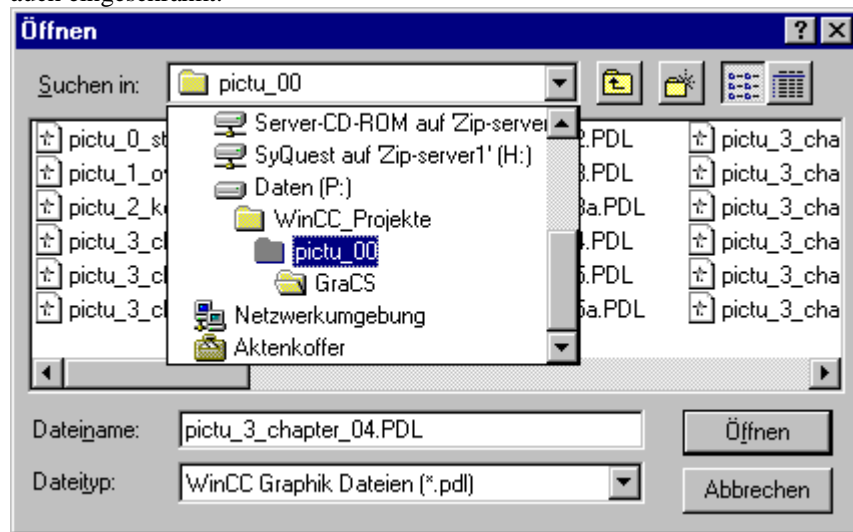
Nachfolgend ein Auszug aus dem Projekt für die Bildprojektierung.

Inhalt von 'P:\WinCC_Projekte\pictu_00\GraCS'			
Name	Größe	Typ	Geändert am
pictu_0_startpicture_00.PDL	16 KB	WinCC.Graphics.Do...	23.08.97 15:52
pictu_1_overview_00.PDL	17 KB	WinCC.Graphics.Do...	22.08.97 15:20
pictu_2_keyboard_00.PDL	60 KB	WinCC.Graphics.Do...	22.08.97 18:21
pictu_3_chapter_00.PDL	9 KB	WinCC.Graphics.Do...	15.08.97 17:45
pictu_3_chapter_01.PDL	81 KB	WinCC.Graphics.Do...	12.08.97 13:22
pictu_3_chapter_01a.PDL	33 KB	WinCC.Graphics.Do...	14.08.97 08:07
pictu_3_chapter_02.PDL	33 KB	WinCC.Graphics.Do...	19.08.97 10:17
pictu_3_chapter_03.PDL	27 KB	WinCC.Graphics.Do...	22.08.97 17:25
pictu_3_chapter_03a.PDL	23 KB	WinCC.Graphics.Do...	14.08.97 16:20
pictu_3_chapter_04.PDL	32 KB	WinCC.Graphics.Do...	18.08.97 18:18
pictu_3_chapter_05.PDL	127 KB	WinCC.Graphics.Do...	11.08.97 12:35
pictu_3_chapter_05a.PDL	111 KB	WinCC.Graphics.Do...	15.08.97 17:52
pictu_3_chapter_05b.PDL	138 KB	WinCC.Graphics.Do...	17.08.97 15:41
pictu_3_chapter_06.PDL	78 KB	WinCC.Graphics.Do...	21.08.97 13:23
pictu_3_chapter_06a.PDL	68 KB	WinCC.Graphics.Do...	21.08.97 22:29
pictu_3_chapter_07.PDL	29 KB	WinCC.Graphics.Do...	18.08.97 18:19

Die zweite Art der Bildübernahme funktioniert über das Öffnen einer Bilddatei (*bild.pdl*) im *Graphics Designer* über den Menüpunkt *Datei* → *Datei öffnen*. Anschließend wird das Bild im aktuellen Bildverzeichnis (\GraCS) über den Menüpunkt *Datei* → *Speichern unter* abgelegt. Diese Vorgehensweise eignet sich dann, wenn die Bilddateien als Basis verwendet werden und sofort eine Anpassung durchgeführt werden soll.

Einschränkung

Bei Kurvenfenstern, Meldefenstern, Tabellenfenstern funktioniert die Übernahme, wenn auch eingeschränkt.



Hinweis:

Die übernommenen Referenzen zu Objekten aus dem bisherigen Projekt stimmen nicht mehr überein. Die Definition der zugehörigen Referenzen im neuen Projekt ist erforderlich.

Diese Referenzen können sein:

Referenzen in Bildern

- Strukturen aus dem Bereich Datentypen
- Interne oder externe Variablen
- Systemvariablen
- Meldefenstervorlagen
- Archivfenstervorlagen (Kurven- oder Tabellenfenster für Prozeßwert- oder Anwenderarchive)
- Bildobjekte, die als Bitmap oder Metafile abgelegt sind (z.B. für Zustandsanzeigen oder Grafikobjekte)
- Weitere Bilder, die als Grafik- oder Prozeßboxen bzw. eingeblendete Fenster genutzt werden
- Verwendete Projektfunktionen
- Zugriffsrechte

Folgende Referenzen müssen zusätzlich definiert werden:

- Definition der Strukturen unter Datentypen, z.B. Regler- oder Vorlagenstrukturen für Anwenderobjekte
- Definition der Kommunikationskanäle und logischen Verbindungen mit der Definition der Variablen (evtl. mit Variablengruppen)
- Definition der internen Variablen bzw. Systemvariablen (mit @ beginnend)

- Definition neuer bzw. ähnlicher Fenstervorlagen und Verbinden der Applikationsfenster (für *Alarm Logging* und *Tag Logging*)
- Übernahme der Bildelemente (*.bmp bzw. *.emf) durch Kopieren aus dem Verzeichnis \GraCS
- Übernahme von Bildfensterinhalten durch Kopieren der weiteren Bilddateien (*.pdl) aus dem Verzeichnis \GraCS
- Die verwendeten Projektfunktionen müssen vom Quellprojekt in das Verzeichnis \Library des neuen Projektes kopiert werden. Zusätzlich muß im Editor *Global Script* über *Header generieren* die Hinterlegung dieser Funktionen im Funktionsbaum erfolgen. Diese Vorgehensweise wurde bereits detaillierter im Kapitel 4.1 *Die Entwicklungsumgebung für Scripte in WinCC* beschrieben.
- Festlegung von verwendeten Zugriffsrechten im Editor *User Administrator*. Die verwendeten Zugriffsrechte (z.B. bei Bedientasten) müssen für die Gruppenfestlegungen definiert werden.

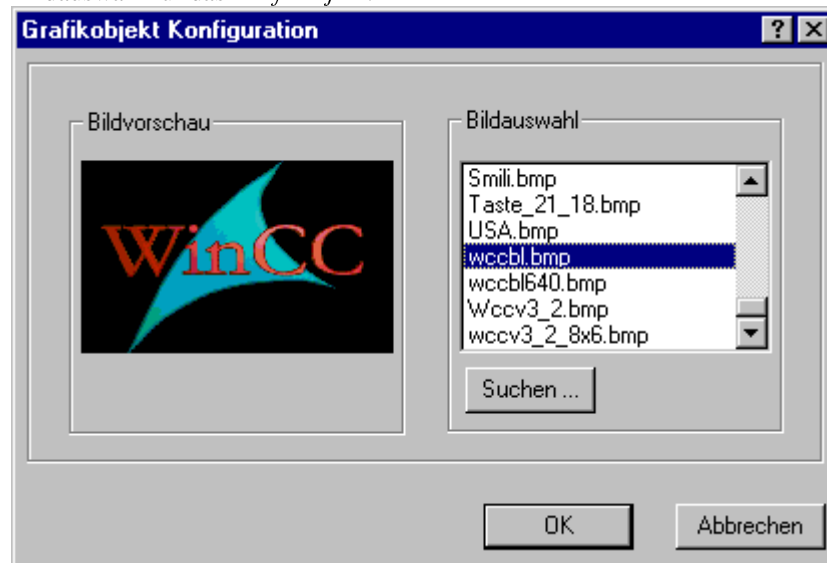
3.3.9.2 Übernahme von Symbolen und Bitmaps

Kopieren

Symbole für *Zustandsanzeigen* oder *Grafikobjekte* in Bilddateien werden als eigene Dateien in dem Bildverzeichnis des Projektes hinterlegt. Kopieren Sie dazu die gewünschten Symboldateien (*.emf oder *.bmp) in das Zielverzeichnis \GraCS des neuen Projektes. Nun sind diese Bilder sofort in der Auswahlliste bei *Zustandsanzeigen* oder *Grafikobjekten* (siehe Objektpalette im *Graphics Designer*) verfügbar. Ausschnitt aus dem Konfigurationsdialog der *Zustandsanzeige*.



Bildauswahl für das *Grafikobjekt*.



Importieren

Symbole können entweder über den eben beschriebenen Weg in ein Bild eingebunden werden oder können direkt über den Menüpunkt *Einfügen* → *Importieren* in das in Bearbeitung befindliche Grafikbild kopiert werden. In diesem Fall muß keine Datei kopiert werden, sondern Sie importieren das gewünschte Symbol direkt über den Zugriff auf den Pfad des Quellprojektes (*\GraCS*) und der gewünschten Symboldatei (*.bmp, *.emf, *.wmf). Das Symbol erscheint nach dem Import direkt als Objekt im Bild (oben links).

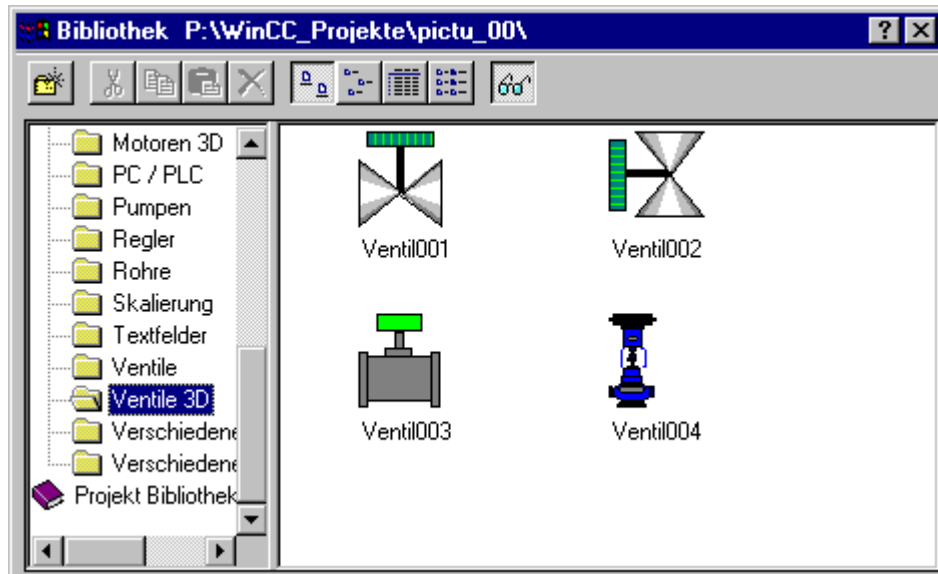
Werden Symbole in einer Projektbibliothek hinterlegt, so kann die Projektbibliothek in einem weiteren Projekt durch die Übernahme der gesamten Projektbibliothek genutzt werden. Siehe dazu die nachfolgende Beschreibung.

3.3.9.3 Übernahme einer Projektbibliothek (mit vorprojektierten Symbolen und Anwenderobjekten)

Globale Bibliothek

Werden Symbole in einer Projektbibliothek abgelegt, so kann diese in einem weiteren Projekt durch kopieren der Datei *library.pxl* in den Pfad *\Library* weiter genutzt werden.

Die vorprojektierten Bausteine können nun jederzeit im neuen Projekt weiter verwendet werden:


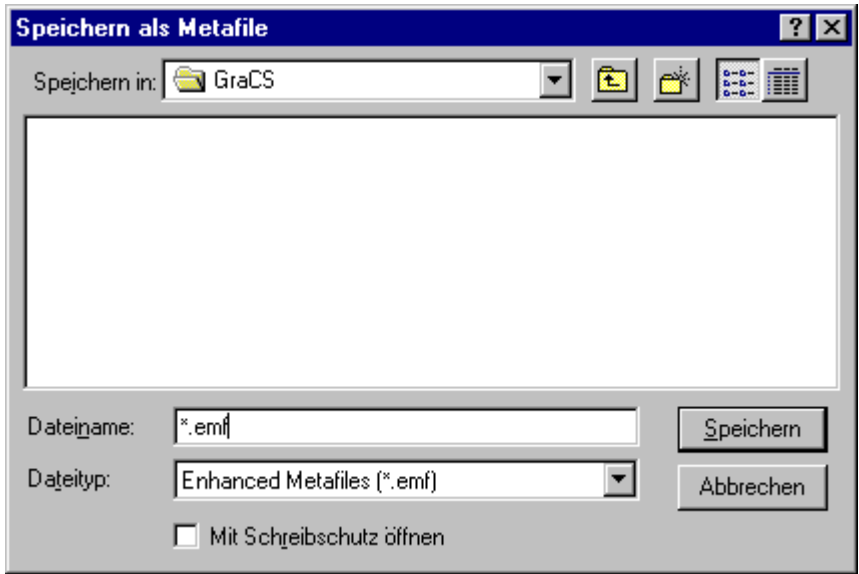


Hinweis:

Berücksichtigen Sie dabei, daß verbundene Symbole eventuell auf nicht vorhandene Referenzen (oder Variablen) verweisen, die Sie erst definieren müssen. Abhängig von der Projektierung dieser Symbole müssen die dazugehörigen Aktionen oder Verbindungen eventuell angepaßt werden. Prüfen Sie daher nach der Verwendung von Symbolen aus der Bibliothek, welche Eigenschaften/Ereignis-Verbindungen bereits vorliegen und ob diese eventuell angepaßt werden müssen.

Einzelne Symbole

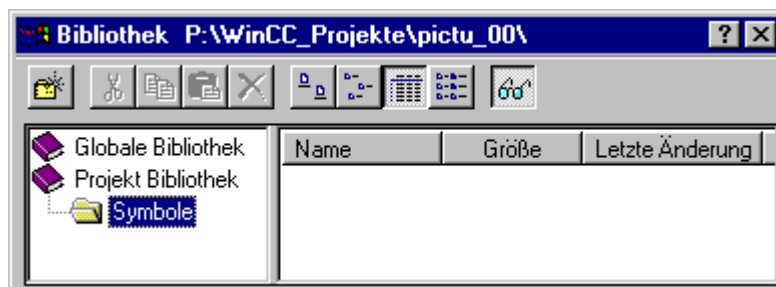
Werden nur einige bestimmte Symbole aus der Projektbibliothek im neuen Projekt genutzt, so werden diese Symbole einzeln exportiert (Symboldatei *.emf).

Schritt	Vorgehen: Symbole übernehmen
1	Bibliothek öffnen.
2	Das gewünschte Symbol mit der  auswählen und bei gedrückter Maustaste das Symbol in das Bild ziehen (Drag&Drop).
3	Mit <i>Datei</i> → <i>Export...</i> den Dialog zum Speichern des Symbols aufrufen. <div data-bbox="486 560 1348 1131" style="border: 1px solid gray; padding: 5px; margin: 5px 0;">  </div>
4	Das Symbol speichern.

Neue Projektbibliothek

Diese exportierten Symbole stehen nun als einzelne Symboldateien zur Verfügung und können über den Weg des Imports einzeln genutzt werden. Werden diese Symbole im Projekt häufiger verwendet, so sollten diese Symbole wieder in die neue Projektbibliothek eingehängt werden. Rufen Sie dazu die Symbolbibliothek, speziell die Projektbibliothek auf.

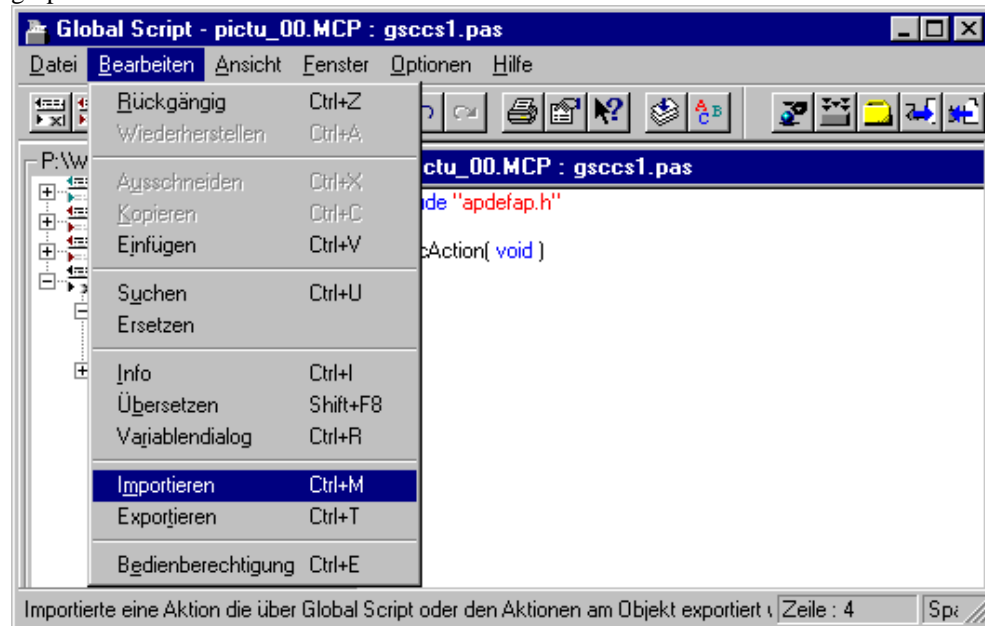
Legen Sie ein eigenes Symbolverzeichnis, z.B. mit Hilfe des Verzeichnissymbols in der Symbolleiste des Bibliotheksfensters an und kopieren Sie mittels *Drag&Drop* die importierten Symbole in dieses Verzeichnis. Auf diesem Wege können Sie teilweise Ihre Symbole aus Projekten übernehmen und weitere spezielle Symbole hinzufügen, damit wieder eine eigene projektspezifische Bibliothek entsteht.



3.3.9.4 Übernahme von Aktionen

Aktionen, die in einem Projekt häufiger benötigt werden oder von einer Objektaktion an eine andere Objektaktion kopiert werden sollen, werden als eigene Dateien abgelegt. Diese Dateien werden im Verzeichnis `\GraCS` mit der Erweiterung `.act` gespeichert. Diese Dateien können durch Kopieren vom Quellverzeichnis in das Zielverzeichnis jederzeit übernommen werden.

Eine Aktionsdatei wird über den Editor für *C-Aktionen* mittels der Taste *Aktion exportieren* in der Symbolleiste in die selbst benannte Zielformatdatei (mit der Endung `.act` für Aktion) gespeichert.



Mit der Taste *Aktion importieren* wird eine gespeicherte Aktionsdatei für eine Objektaktion im Bild des neuen Projektes übernommen. Beachten Sie dazu auch die Beschreibung in Kapitel 4.1 *Die Entwicklungsumgebung für Scripte in WinCC*.

Hinweis:

Häufiger verwendete Aktionen können auch als Projekt- oder Standard-Funktionen definiert werden können.

3.3.9.5 Übernahme von Variablen

Der Variablenhaushalt von WinCC kann über mehrere Wege ergänzt werden:

- Einlesen von S5-Datenvariablen bzw. S7-Datenvariablen über Assistenten (*Dynamic Wizard*)
- Übernahme von *S7-Variablen* mittels des *PCS7-Mappers*
- Import und Export von Textlisten mittels des Zusatztools *Var_Exim*
- Interaktiver Zugriff auf die Datenbanktabellen (Variablen tabellen)
- Eigens programmierte Assistenten (*Dynamic Wizard*) oder Programme, die mittels der WinCC API-Funktionen neue Daten im Variablenhaushalt erzeugen


Die letzten zwei aufgeführten Möglichkeiten setzen sehr gute Kenntnisse hinsichtlich des Umganges mit SQL-Datenbanken sowie der Programmierung über die

Applikationsschnittstelle voraus. Dies sollte nur von Personen mit solchen Kenntnissen durchgeführt bzw. vorbereitet werden.

Vor der Übernahme der Daten in das Zielprojekt muß daher geklärt werden, wo die Basis für das Zielprojekt liegt. Liegt bereits eine größere Anzahl von Variablen im Variablenhaushalt von WinCC vor, so sollte die Variablenliste von WinCC als Import für das Zielprojekt herangezogen werden. Die *internen Variablen* müssen auf jeden Fall aus dem *Variablenhaushalt* von WinCC übernommen werden. Dazu muß das Tool *Var_Exim.exe* eingesetzt werden.

S5-/S7-Datenvariablen mit Dynamic Wizard übernehmen

Die mit STEP5/STEP7- Software erstellten Datenbereichsdefinitionen können in den Variablenhaushalt von WinCC mit Hilfe der vorhandenen *Dynamic Wizards* eingelesen werden. Folgende Tätigkeiten sind durchzuführen:

Schritt	Vorgehen: Daten von S5 oder S7 übernehmen
1	Eine Datensicherung des Projektes durchführen. Es werden Veränderungen in der Datenbank vorgenommen.
2	Mit der STEP-Software die Zuordnungsliste exportieren. Es entsteht eine Datei <i>prj_zuli.SEQ</i>
3	Diese exportierte Datei von speziellen Symbolen (z.B. zu Programmaufrufen) bereinigen, die für den Import in WinCC nicht benötigt werden. Dies kann mit Hilfe eines Texteditors (z.B. Wordpad) durchgeführt werden. Die Zuordnungsliste darf keine Leerzeilen enthalten.
4	Das Ziel-Projekt im WinCC <i>Control Center</i> öffnen. Das Projekt muß sich im Projektierungsstatus befinden (Runtime nicht aktiv).
5	Den Editor <i>Graphics Designer</i> öffnen. In einem beliebigen Bild im <i>Dynamic Wizard</i> (einblenden über <i>Ansicht</i> → <i>Symbolleisten...</i>) die Registerkarte <i>Import-Funktionen</i> wählen. Dort die Funktion <i>Import S7 – S5 ZULI</i> wählen. Im folgenden müssen Sie die <i>Quell-Datei (.seq)</i> mit Verzeichnisangabe nennen (über den Button ). Weiters ist die logische Verbindung anzugeben, in die Variablenbeschreibungen der Zuordnungsliste eingehängt werden sollen. Die Daten werden nun in den WinCC- Variablenhaushalt eingetragen. Die Variablennamen müssen im gesamten WinCC-Projekt eindeutig sein. Die Variablen werden dem evtl. bereits bestehenden WinCC-Variablenhaushalt hinzugefügt. Der Variablenname ist dabei der Schlüssel.

Variablenübernahme mit dem Hilfsprogramm

Die Verbindungen (Kanal-DLL, logischer Verbindung sowie Verbindungsparameter) müssen vor dem Import in dem Zielprojekt bereits definiert sein oder zuvor definiert werden.


Hinweis:

Für eine automatische Erzeugung von Verbindungen sowie Dateneintragen in der WinCC-Datenbank könnte ein spezielles Programm erstellt werden, das über die WinCC API-Schnittstelle derartige Definitionen automatisch vornimmt. Über diesen Weg können bereits bestehende Anlagendaten automatisch ergänzt werden. Dies muß von einem Fachmann für die WinCC API-Programmierung bzw. SQL-Programmierung erstellt werden.

Die im Variablenhaushalt definierten Variablen können jederzeit als Textdatei **exportiert** werden, um die Liste der Variablen zu ergänzen. Anschließend müssen diese erzeugten Daten wieder in den Variablenhaushalt des Projektes **importiert** werden. Die erzeugten Dateien sind im Format CSV (comma separated values) abgelegt und können von beliebigen Aufbereitungsprogrammen gelesen und weiterverarbeitet werden. Dazu existiert eine eigens Anwendungsprogramm auf der WinCC CD-ROM unter dem Verzeichnis `\SmartTools\CC_VariablenImportExport`. Dieses Windows Programm wird als Zusatzprogramm bereitgestellt:

- zur Auslagerung der Daten des Variablenhaushalts
- für den Import von bereits extern erstellten Variablendaten
- für die Massendaten-Projektierung

Für den Export bzw. Import der Daten müssen nun folgende Schritte durchlaufen werden:

Schritt	Vorgehen: Variablen importieren, exportieren
1	Öffnen Sie Ihr WinCC-Projekt im WinCC <i>Control Center</i> .
2	Definieren Sie nicht vorhandene, aber für den späteren Import nötige Verbindungen (Kanal-DLL – logische Verbindung – Verbindungsparameter). Dies sollte aber erst im neuen Projekt erfolgen. Hierzu ist evtl. ein zweiter Export-Import-Vorgang nötig.
3	Aktivieren Sie das Programm <i>Var_exim</i> über  D. Die Benutzeroberfläche dieses Programms ist nachfolgend dargestellt.

Import - Export

Für den Export bzw. Import sind folgende Einstellungen notwendig:

Ort, Aktion	Import	Export
<i>Path</i>	Das Projektverzeichnis mit den Dateien für den	Das Projektverzeichnis für den Variablenexport angeben. Die Auswahl

Ort, Aktion	Import	Export
	Variablenimport auswählen. Die Auswahl erfolgt durch Selektion der Datei <i><projektname>.mcp</i> . Die Dateien mit den Daten für den Import müssen im selben Verzeichnis liegen wie die Projektdatei.	erfolgt durch Selektion der Datei <i><projektname>.mcp</i> .
Aktion	<i>Import</i> anwählen. Sollen bereits vorhandene Variablen Daten überschrieben werden, ist <i>Import overwrite</i> anzuwählen.	<i>Export</i> anwählen.
<i>Execute</i>	Ausführen wählen Der anschließende Dialog zeigt die eingestellten Parameter und führt, nach Bestätigung mit <i>OK</i> , die Umsetzung durch. Aufgrund der durchgeführten Überprüfungen nimmt der Import mehr Zeit in Anspruch.	Ausführen wählen Der anschließende Dialog zeigt die eingestellten Parameter und führt, nach Bestätigung mit <i>OK</i> , die Umsetzung durch.
Statusanzeige	End Export/Import	End Export/Import
Variablendatei <i>Name_vex.csv</i>	Basis für den Import: Besteht aus Kopfzeile und Datensätzen	Die erzeugte Variablenliste wird in Textform in dieser Datei abgelegt. Diese kann entweder direkt über die Taste <i>var</i> geöffnet werden oder mit einem Texteditor (Notepad) oder mit EXCEL bearbeitet werden.
Variablendatei <i>Name_cex.csv</i>	Basis für den Import: Besteht aus Kopfzeile und Datensätzen (Strukturkomponenten)	In dieser Datei stehen die aktuell konfigurierten Verbindungen, auf die in der Variablendatei Bezug genommen wird. Diese kann entweder direkt über die Taste <i>con</i> geöffnet werden oder mit einem Texteditor (Notepad) oder mit EXCEL bearbeitet werden.
Datenstrukturdatei <i>Name_dex.csv</i>	Basis für den Import: Besteht aus Kopfzeile und Datensätzen	Falls Variablen mit Datenstrukturtypen enthalten sind, entsteht zusätzlich diese Datei mit Strukturinformationen. Ihr Inhalt kann über einen Texteditor (Notepad) oder mit EXCEL bearbeitet werden.
Diagnosedatei <i>Diag.txt</i>	Diagnosedatei mit Hinweisen, welche Variablen nicht importiert werden konnten.	

Mit der Taste *End* kann das Programm verlassen werden.

Variablenliste

In der folgenden Tabelle ist der Aufbau der Variablenlisten beschrieben.

Feld	Typ	Beschreibung
Varname	char	Variablenname
Conn	char	Verbindung
Group	char	Gruppenbezeichnung
Spec	char	Interne Variable oder Adresse (passend zur Verbindungsart)
Flag	DWORD	
Common	DWORD	
Ctyp	DWORD	Variablentyp 1 BIT 2 SBYTE 3 BYTE 4 SWORD 5 WORD 6 SDWORD 7 DWORD 8 FLOAT 9 DOUBLE 10 TEXT_8 11 TEXT_16 12 Rohdatentyp 13 Feld 14 Struktur 15 BITFELD_8 16 BITFELD_16 17 BITFELD_32 18 Textreferenz
CLen	DWORD	Länge der Variablen
CPro	DWORD	Interne oder externe Variable
CFor	DWORD	Formatkonvertierung
Protocol		
P1	BOOL	Fehler in Obergrenze
P2	BOOL	Fehler in Untergrenze
P3	BOOL	Umwandlungsfehler
P4	BOOL	Schreibfehler
P5	BOOL	
P6	BOOL	
L1	BOOL	Ersatzwert bei Obergrenzenfehler
L2	BOOL	Ersatzwert bei Untergrenzenfehler
L3	BOOL	Startwert
L4	BOOL	Ersatzwert bei Verbindungsfehler
L5	BOOL	Obergrenze gültig
L6	BOOL	Untergrenze gültig
L7	BOOL	Startwert gültig

Feld	Typ	Beschreibung
L8	BOOL	Ersatzwert gültig
LF1	double	Obergrenze
LF2	double	Untergrenze
LF3	double	Startwert
LF4	double	Ersatzwert
Skalierung		
SCF	DWORD	1 wenn Skalierung definiert ist
SPU	double	Wertebereich Prozeß von
SPO	double	Wertebereich Prozeß bis
SVU	double	Wertebereich Variable von
SVO	double	Wertebereich Variable bis

Verbindungsliste

Feld	Typ	Beschreibung
Conname	char	Logischer Verbindungsname
Unit	char	Kanal Unit
Common	char	Allgemein
Specific	char	Spezifische Verbindungsparameter
Flag	DWORD	

Datenstrukturliste

Feld	Typ	Beschreibung
Datastructure	short	Datenstrukturname oder Komponentename
Type ID	short	Identifikation (wird in Variablenliste unter Ctyp verwendet)
Creator ID	short	

Um die Textlisten in EXCEL (Version 7.0 oder 8.0) weiterverarbeiten zu können, müssen Sie die exportierten Dateien mit dem Dateityp *Textdateien* [**.prn; *.txt; *.csv*] öffnen.

Vorschriften

Die Anpassung der Variablendaten in der Textliste erfolgt unter folgenden speziellen Vorschriften:

Typ	in Textliste	in WinCC
Verbindungen	allgemeine Beschreibung	Logische Verbindungen müssen , falls nicht vorhanden, neu definiert werden!
	Spezielle Beschreibung Kanal-DLL	Kanal-DLL müssen , falls nicht vorhanden, neu definiert werden!
Variablen- gruppe	Keine Gruppeninformation Sind im Projekt Gruppen definiert, die keine Variable enthalten, so werden diese Gruppen auch nicht exportiert.	Die Gruppeninformation wird beim Import mit der ersten Variablen einer Gruppe automatisch generiert.
Variablen	Allgemeine Beschreibung	
	Spezifische Beschreibung jeweilige Kanal-DLL oder interne Variable	Kanal-DLL oder interne Variable
	Beim Export werden die fehlenden Anteile durch * ersetzt.	Variablen mit fehlender spezifischer Beschreibung werden nicht importiert!
Variablen vom Typ Daten- struktur	Zuordnung entsprechend der Datenstrukturdefinition der Strukturliste	Wird dem Datentyp zugeordnet.
Datenstruktur Definition		Muß vom Projektteur definiert werden.
Grenzen	Werden nicht über die Textliste ex- oder importiert	Muß vom Projektteur definiert werden.

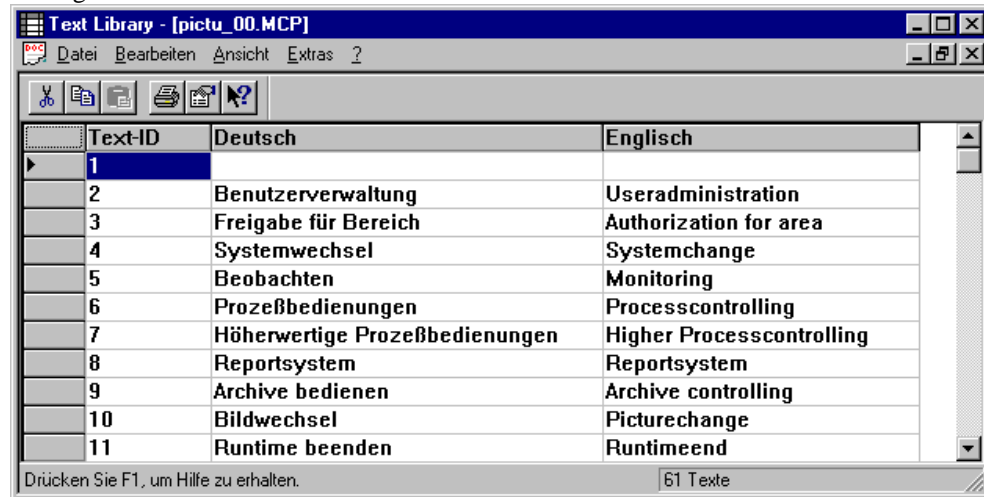
Bevor Sie den Import der geänderten bzw. neuen Variablen vornehmen, führen Sie zuerst eine **Datensicherung** Ihres Projektes durch. Es werden Veränderungen in der Datenbank vorgenommen. Diese Datenbankveränderungen können in WinCC nicht rückgängig gemacht werden.

3.3.9.6 Übernahme von mehrsprachigen Texten (aus Bildern, in Meldungen)

Mehrsprachige Texte der Bilder

Die in den Bildern hinterlegten mehrsprachigen Texte können durch das Kopieren des Bildes selbst in das neue Projekt übernommen werden.

In dem neuen Projekt muß die jeweilige Sprache in der Textbibliothek angefügt werden. Prüfen Sie dazu die Einstellungen in der Text Library. Für jede Sprache muß eine Spalte vorliegen!



Sollen in dem neuen Projekt die Sprachen nur teilweise übernommen werden? Da die gesamte Textinformation pro Bild hinterlegt wird, muß pro Bild eine Nachprojektierung in der jeweiligen Sprache vorgenommen werden. Es ist dabei zu überlegen, ob es sinnvoll ist, die bereits projektierten Texte wieder zu entfernen. Das Umschalten zur Laufzeit wird nur über speziell projektierte Tasten ermöglicht und könnte somit für das Projekt eingeschränkt werden. Sollen die Texte für eine bereits eingeführte Sprache trotzdem entfernt werden, so wird der Export und Import über das Zusatz-Tool *language* empfohlen.

Bilder mit Textreferenzen übernehmen

Werden in den übernommenen Bildern Textreferenzen genutzt, müssen folgende Daten mit übernommen werden:

- Übernahme der zugehörigen Variablen (Export oder Neudefinition) aus dem Variablenhaushalt des WinCC-Projektes
- Übernahme der Texte aus der Textbibliothek
- Die Textreferenzvariablen müssen mit den gültigen Text-Identifikationsnummern (Text-ID's) versehen sein. Prüfen Sie, ob die Text-ID's noch mit den zugehörigen Texten übereinstimmen.

Übernahme von Texten aus der Textbibliothek

Werden die Texte aus der Textbibliothek nur teilweise übernommen, so muß eine Anpassung der Text-IDs vorgenommen werden. Die Übernahme von Texten aus der Textbibliothek ist über den Export/Import-Mechanismus im Editor *Text Library* möglich.

3.3.9.7 Übernahme von Meldungen

Die Informationsgrundlage für Meldungen (Alarmer) ist

- änderungsbedürftig und
- aufgrund der Menge (Massendaten) projektierungsaufwendig.

Aus diesem Grunde wird die Übernahme von Meldedaten aus bisherigen Projekten sehr häufig genutzt. Abhängig von der Quelle der Meldedaten sind folgende Übernahmewege möglich:

- Übernahme der bereits projektierten Meldeinformationen aus Vorgänger-Systemen (z.B. COROS)
- Import von (Einzel-)Meldungen aus einem existierenden WinCC-Projekt
- Import von Meldeinformationen aus der Konzeptphase

Übernahme von Meldungen aus COROS

Die Vorgehensweise für die aufgeführten Quellen sieht folgendermaßen aus:

Schritt	Vorgehen: Übernahme vorhandener COROS Meldungstexte
1	In COROS Meldeinformationen exportieren (<i>mldtexte.txt</i>)
2	In WinCC über den <i>Dynamic Wizard Importfunktionen</i> → <i>Import Meldungen</i> diese Meldedatei einlesen. Die Daten werden nun in das aktuelle WinCC-Projekt importiert.

Übernahme von Meldungen aus einem WinCC-Projekt

Werden Meldungen aus einem vorhandenen Projekt übernommen, so muß zunächst geklärt werden, ob das Ziel-Meldesystem in der Datenbankstruktur in der selben Form aufgebaut wurde oder nicht. Die Unterschiede sind z.B. in den Anwender- und Prozeßwertblöcken zu sehen. Organisieren Sie die Ziel-Datenblöcke in der Reihenfolge (als auch in der Länge der Textelemente) möglichst gleich. Ansonsten müssen Sie vor dem Import eine Anpassung in den einzelnen Spalten vornehmen.

Schritt	Vorgehen: Übernahme vorhandener WinCC Meldungstexte
1	Öffnen Sie den Editor <i>Alarm Logging</i> im aktuellen Projekt.
2	Rufen Sie den Export der Meldungen über den Menüpunkt <i>Meldungen</i> → <i>Einzelmeldungen exportieren</i> auf.
3	Nennen Sie die Ziel-Textdatei, in welche die zu exportierenden Meldeinformationen abgelegt werden sollen. Selektieren Sie über <i>Selektion</i> die zu exportierenden Meldungen über die Kriterien z.B. Meldenummer, Meldeklasse.
4	Starten Sie den Exportvorgang über <i>Export</i> . Es entsteht nun eine Textdatei mit Meldeinformationen, die jeweils mit Komma getrennt sind.
5	Schließen Sie das aktuelle Projekt und öffnen Sie das neue Projekt. Öffnen Sie wieder den Editor <i>Alarm Logging</i> und definieren Sie zunächst die nötigen Meldeklassen und Meldearten. Pro Meldeart definieren Sie eine Meldung, um das Rohgerüst für den folgenden Import zu erhalten.
6	Für den Export dieser Basismeldungen rufen Sie <i>Meldungen</i> → <i>Einzelmeldungen exportieren</i> auf. Gehen Sie entsprechend Schritt 3 und 4 vor.
7	Öffnen Sie nun z.B. in EXCEL die Meldedatei des Quellprojektes und die Meldedatei des Zielprojektes. Die Spalten sind jeweils mit Komma getrennt. Vergleichen Sie den Aufbau der Meldeblöcke und nehmen Sie eventuell Anpassungen durch umsetzen oder umbenennen von Spalten vor. Tragen Sie in den Blöcken mit den Text-IDs jeweils den Index 0 ein. Über diesen Weg werden die Texte durch den Import automatisch in der Textbibliothek organisiert. Die alten Ident-Nummern dürfen auf keinen Fall beibehalten werden! Die geänderte Datei muß erneut als Textdatei gespeichert werden
8	Rufen Sie nun den Importvorgang über <i>Meldungen</i> → <i>Einzelmeldungen importieren</i> auf.
9	Nennen Sie nun die Quell-Textdatei mit den exportierten Meldeinformationen. Im folgenden müssen Sie entscheiden, ob beim Import bereits vorhandene Meldungen überschrieben werden sollen. Die Zuordnung erfolgt über die Meldenummer, die im Projekt eindeutig definiert sein muß.
10	Die Meldungen werden anschließend importiert und ergänzen Ihr vorhandenes Meldesystem mit den bereits projektierten Meldeinformationen. Prüfen Sie die importierten Zuordnungen.

**Hinweis:**

Werden Meldedaten aus einem WinCC-Projekt der V. 1.10 übernommen, müssen die Spaltenüberschriften in der Meldetextdatei beachtet werden!

Import von Meldeinformationen aus der Konzeptphase mittels EXCEL-Tabellen

Die Meldeinformationen liegen in einer EXCEL-Tabelle bereits vor. Durch Übernahme der Spalten in die Meldestruktur des WinCC-Projektes können diese Meldungen übernommen werden. Dazu muß auf einer WinCC Meldedatei aufgebaut werden. Diese wird folgendermaßen erzeugt:

Schritt	Vorgehen: Meldestruktur anlegen
1	Öffnen Sie das neue Projekt im WinCC <i>Control Center</i> . Öffnen Sie den Editor <i>Alarm Logging</i> und definieren Sie zunächst die nötigen Meldeblöcke, Meldeklassen und Meldearten. Pro Meldeart definieren Sie eine Meldung, um das Rohgerüst für den folgenden Import zu erhalten.
2	Für den Export dieser Basismeldungen rufen Sie <i>Meldungen</i> → <i>Einzelmeldungen exportieren</i> auf.
3	Nennen Sie die Ziel-Textdatei, in welche die zu exportierenden Meldeinformationen abgelegt werden sollen.
4	Starten Sie den Exportvorgang über <i>Export</i> . Es entsteht nun eine Textdatei mit den Meldeinformationen, die durch Komma getrennt sind.
5	Öffnen Sie in EXCEL die Meldedatei und die neu erzeugte Meldedatei des Ziel-Projektes. Die Spalten sind durch Komma getrennt. Legen Sie jeweils eine kopierte Zeile zu der entsprechenden Meldeklasse/Meldeart in der Tabelle an. Übernehmen Sie aus den Quelldaten die Meldetexte, etc. und tragen Sie diese in den zugehörigen Blöcken ein. Zum Beispiel: <i>Block 1</i> → <i>Meldetext</i> . Nummerieren Sie alle Meldezeilen (z.B. von 1 ab) durch. Dies kann in EXCEL sehr schnell mit Hilfe der Aufzählungen in der Meldenummernspalte erzeugt werden.
6	Tragen Sie in den Blöcken mit den Text-ID's jeweils den Index 0 ein. Über diesen Weg werden die Texte durch den Import automatisch in der Textbibliothek organisiert. Die alten Ident-Nummern dürfen auf keinen Fall beibehalten werden!! Die geänderte Datei muß wiederum als Textdatei gespeichert werden.
7	Rufen Sie im Alarm Logging Editor den Importvorgang über <i>Meldungen</i> → <i>Einzelmeldungen importieren</i> auf.
8	Nennen Sie die Quell-Textdatei mit den exportierten Meldeinformationen. Im folgenden definieren Sie die Parameter so, daß bereits vorhandene Meldungen überschrieben werden. Die Zuordnung erfolgt über die Meldenummer, die im Projekt eindeutig sein muß.
9	Die Meldungen werden nun importiert und ergänzen Ihr vorhandenes Meldesystem mit den bereits projektierten Meldeinformationen. Prüfen Sie die importierten Zuordnungen.

3.3.9.8 Übernahme von Meßwerten

Da die Festlegung von Meßpunkten sowie die Definition der Prozeßwertarchive als auch der Anwenderarchive mit Ihren Eigenschaften direkt in die Datenbankstruktur integriert sind, ist eine Übernahme (ohne direkte Datenbankzugriffe mittels fundierter Datenbankkenntnisse) nicht möglich. Das heißt, daß eine Neuprojektierung dieser Archive und Meßpunkte erfolgen muß oder zu Beginn der Projektierung die Daten aufgrund der Kopie eines gesamten Basis-Projektes automatisch übernommen werden.

3.3.9.9 Übernahme von Drucklayouts

Kopieren Sie die gewünschten Drucklayouts, *.rpl bei Seitenlayouts oder *.rpl bei einem Zeilenlayout, aus dem Quellverzeichnis in das Verzeichnis \PRT des neuen Projektes.

3.3.9.10 Übernahme von globalen Aktionen

Kopieren Sie die gewünschten globalen Aktionen oder Hintergrundaktionen *.pas aus dem Quellverzeichnis in das Verzeichnis \Pas des neuen Projektes.

3.3.9.11 Übernahme von Projektfunktionen

Kopieren Sie die gewünschten Projektfunktionen *.fct aus dem Quellverzeichnis in das Verzeichnis \Library des neuen Projektes. Um diese Funktionen im Projekt bekanntzugeben, muß im Editor *Global Scripts* der Menüpunkt *Optionen* → *Header generieren* aktiviert werden. Eine detaillierte Beschreibung hierzu finden Sie im Kapitel 4.1 *Die Entwicklungsumgebung für Scripte in WinCC*.

3.3.9.12 Nutzen von Standard-Funktionen

Standard-Funktionen müssen im Gegensatz zu *Projekt-Funktionen* nicht zusätzlich kopiert werden. Diese Funktionen sind sofort für das Projekt verfügbar, da diese Funktionen für alle WinCC-Projekte auf der Station bekannt sind.

3.3.9.13 Übernahme der Benutzerverwaltung

Da die Festlegungen für Benutzergruppen, Benutzer und Zugriffsrechte direkt in die Datenbankstruktur integriert sind, ist eine Übernahme nicht möglich. Aus diesem Grunde ist eine Neuprojektierung erforderlich.

Eine Ausnahme bietet zu Projektierungsbeginn eine automatische Datenübernahme aufgrund der Kopie eines gesamten Basisprojektes.

3.3.10 Mauslose Bedienung

Die Bedienung der Anlagenbilder erfolgt unter WinCC in vielen Fällen über die Maus. Der Mausclick ist das Ereignis unter den jeweiligen Dynamisierungen, der am häufigsten und in den unterschiedlichsten Varianten (Mausclick links bzw. rechts drücken oder loslassen) eingesetzt wird. Es existieren aber auch Anlagen, die in gemischter Form oder nur mit Tastatur bedient werden. Operator Panels zum Beispiel werden nur über Tastatur bedient.

3.3.10.1 Bedienung über Tastatur

Die Bedienung über Tastatur unterscheidet dabei folgende Eingabemöglichkeiten:

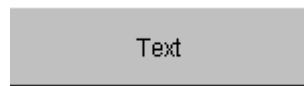
- Funktionstasten F1 bis F12
- Spezielle Funktionstasten (z.B. Operator Panel Funktionstasten SF10)
- Standard Tastatureingaben
- Bewegungen über Eingabefelder bzw. Bedientasten mittels Richtungstasten bzw. über spezielle Tasten

Die Projektierung der Bedienungen ohne Maus muß für folgende Projektierungsbereiche im einzelnen betrachtet werden:

- Bedientasten im Anlagenbild (z.B. für Bildwechsel)
 - über Funktionstasten
 - über spezielle Tasten
 - über Standard Tasten
- beliebige Tastenbedienung
- Bewegung über Bedienobjekte
- Eingabefelder im Anlagenbild
 - Ein-/Ausgabefelder
 - spezielle Eingabeobjekte (Checkbox, ...)
- Alarm Logging (Meldefenster)
 - Bedienungen über Funktionstasten
 - Bedienungen über eigens projektierte Tasten
- Tag Logging (Kurven- oder Tabellenfenster)
 - Bedienungen über Funktionstasten
 - Bedienungen über eigens projektierte Tasten
- Druckauftrag über Taste anstoßen
- an- oder abmelden über Tastatur

Bedientasten werden im typischen *Windows-Stil* projektiert. Aus diesem Grunde finden Sie die Standard Windows-Bedientaste in der Objektpalette vor. Diese Taste kann jederzeit über weitere grafische Elemente ergänzt werden.

Bedientasten



Bedienung über Funktionstasten

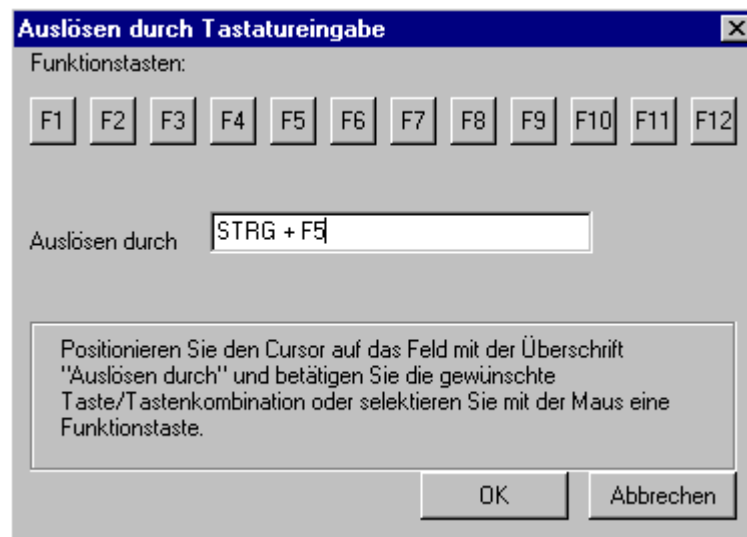
Die Funktionstasten F1 bis F12 auf der Standard-Tastatur werden häufig als (zusätzliche) Tastaturbedienung für Bedientasten zum Bildwechsel in der Anlagenbildhierarchie genutzt. Diese Funktionstasten können zu den projektierten Windows-Buttons als sogenannte Hotkeys jederzeit projektiert werden. Ein Hotkey legt die Schnellbedienung für die jeweilige Funktion fest.



Als Hotkeys können z.B. die genannten Funktionstasten hinterlegt werden. Diese werden bereits als Auswahlknöpfe in der Projektierungsmaske angeboten.

Wird eine Kombination z.B. mit der UMSCHALT-Taste oder der STRG-Taste benötigt, so muß in dem Einstellungsfeld die gewünschte Tasten-Folge (z.B. UMSCHALT, F2) direkt durch Drücken der jeweiligen Tasten eingegeben werden. Es müssen keine Spezial-Codes eingegeben werden.

Diese gewählte Kombination wird in dem Eingabefeld aufgezeigt.



Diese Tastenwahl wird in den Eigenschaften des Objektes hinterlegt und kann entweder über den Konfigurationsdialog oder über die Eigenschaft *Sonstige* → *Hotkey* direkt modifiziert werden.



Aktion für Hotkey

Die Aktion, die aufgrund der Funktionstasteneingabe (Hotkey-Tastenfestlegung) ausgelöst werden soll, muß unter den Ereignissen des Windows-Buttons hinterlegt werden. Das Ereignis hierfür muß der **Mausklick** sein. Das Ereignis wird beim Loslassen der Maustaste ausgelöst, aber nur dann, wenn sich der Mauszeiger beim Drücken und Loslassen der Maustaste über dem Objekt befindet. Liegt hinter dem Ereignis *Mausklick* keine Aktion, sondern z.B. nur hinter dem (ähnlichen) Ereignis *linke Maus drücken*, wird die Aktion über die Funktionstaste **nicht** ausgelöst! Beachten Sie auch bei der Projektierung, daß die Funktionstaste im Bild nur einmal verwendet werden kann.



Spezielle Funktionstasten

Werden spezielle Funktionstasten, z.B. die Tasten des Operator Panels F13, S1 etc. für die Bildbedienung eingesetzt, so müssen diese Tasten auf Tastenkombinationen umgelegt werden. Die Taste F13 könnte z.B. auf die Kombination UMSCHALT F1 gelegt werden. Die gewählten Tastenkombinationen werden neben der oben beschriebenen Verwendung in der Visualisierung zusätzlich gerätespezifisch definiert. Hierzu finden Sie spezielle Tastatur-Einstellungen abhängig von den jeweilig eingesetzten Geräten vor. Zum Beispiel wird für den Industrie-PC eine Datei *F125.key* für die Einstellungen der Tastencodes zur Verfügung gestellt. In diesen gerätespezifischen Dateien werden die Codes für die Funktionstasten hinterlegt. Nach der Anpassung der Geräte-Tastaturdefinition - pro Funktionstaste der jeweilige hexadezimale Code - sowie Aktivierung der neuen Tastaturcodes können diese Tasten in den Anlagenbildern für die Bildbedienung eingesetzt werden.

Standardtasten

Wird die Aktion nicht auf eine Funktionstasteneingabe bezogen, sondern auf eine ausgewählte Taste der Standard-Tastatur z.B. den Buchstaben *m*, so wird diese Taste als Hotkey im Objekt *Windows-Button* hinterlegt.

Beliebige Tastenbedienung

Das Design der Bedientasten kann auch selbst für die Anlagenbilder entworfen werden. Weitere Bedientasten finden Sie z.B. in der Anwenderbibliothek unter *3D-Tasten* oder *Anwenderobjekte*.

Eigens entworfene Objekte, die nicht auf dem Windows-Button basieren, können nicht über den Hotkey projiziert werden. Alle anderen Objekte müssen über das *Tastenergebnis* des Objektes für die Tastenbedienung projiziert werden. Hierzu gibt es pro Objekt folgende Tastatur-Ereignisse:

- drücken
- loslassen



Dieses Tastenergebnis muß für die mögliche Projektierung der Tastaturbedienung verfügbar sein. Beim Einsatz von vordefinierten Tasten aus der Anwenderbibliothek ist daher zunächst zu prüfen, ob diese Taste sich für die Bedienung ohne Maus eignet. Die Umschalt-Buttons der Anwenderobjekte z.B. sind nicht immer für die Tastaturbedienung freigegeben. Eine Anpassung kann daher nötig sein.

Vorprojizierte Tastenbedienungen (z.B. Blättern in der Bildhierarchie) finden Sie in den Optionspaketen (z.B. *Basic Process Control - Picture Tree Manager* etc.).

Wird eines dieser Objekte als Bedienelement genutzt, so wird die auslösende Taste hinter dem Ereignis *Tastatur - drücken* oder *Tastatur - loslassen* projiziert. Als Aktion kann entweder eine *Direktverbindung* oder eine *C-Aktion* projiziert werden.

Das auslösende Tastenergebnis ist entweder eine

- beliebige Tastenaktion oder eine
- ausgewählte Taste der Standard-Tastatur.

Handelt es sich um eine beliebige Taste, so kann die *Direktverbindung* eingesetzt werden; muß hingegen eine spezielle Tasteneingabe geprüft werden, so muß eine *C-Aktion* eingesetzt werden. Die *C-Aktion* prüft den eingegebenen Tastencode vor der Fortführung der eigentlichen Aktionsfolge zeichenweise:

```
if (nChar == S)
    OpenPicture("Start.pdl");
```

Werden Tastenkombinationen abgefragt, so müssen nach der ersten Tastaturcode-Abfrage weitere Prüfungen erfolgen. Hierzu muß die spezielle Funktion „GetAsyncKeyState“ folgendermaßen eingesetzt werden:

```
#include "apdefap.h"
void OnKeyDown(char* lpszPictureName, char* lpszObjectName, char*
               lpszPropertyName, UINT nChar, UINT nRepCnt, UINT
               nFlags)
{

#pragma code("user32.dll");
int GetAsyncKeyState(int vKey);
#pragma code();

int nRet;

//Abfrage auf F1
if (nChar==112) {
    //Wenn F1 dann Abfrage ob SHIFT-Taste betätigt
    nRet = GetAsyncKeyState(VK_SHIFT);
    //Bit gesetzt für Taste gedrückt
    if (nRet & 0x8000) {
        // Hier können weitere Anweisungen
        // editiert werden, die bei der entsprechenden
        // Tastenkombination ausgelöst werden
    }
    //Wenn F1 dann Abfrage ob CTRL-Taste betätigt
    nRet = GetAsyncKeyState(VK_CONTROL);
    //Bit gesetzt für Taste gedrückt
    if (nRet & 0x8000) {
        // Hier können weitere Anweisungen
        // editiert werden, die bei der entsprechenden
        // Tastenkombination ausgelöst werden
    }
    //Wenn F1 dann Abfrage ob ALTGR-Taste betätigt
    if (nFlags & KF_ALTDOWN) {
        // Hier können weitere Anweisungen
        // editiert werden, die bei der entsprech-//
        // Tastenkombination ausgelöst werden
    }
    //Wenn F1 dann Abfrage ob ALT-Taste betätigt
    nRet = GetAsyncKeyState(VK_MENU);
    //Bit gesetzt für Taste gedrückt
    if (nRet & 0x8000) {
        // Hier können weitere Anweisungen
        // editiert werden, die bei der entsprechenden
        // Tastenkombination ausgelöst werden
    }
} //Ende der Tastaturabfrage
}
```

3.3.10.2 Bewegung über Bedienobjekte (Eingabefelder und Bedienfelder)

Mit der Maus kann jedes bedienbare Objekt direkt angeklickt werden. Die Bedienbarkeit wird jeweils durch den veränderten Mauszeiger visualisiert. Wie können diese Objekte ohne Maus bedient werden?

Alpha- Schaltcursor

Die bedienbaren Objekte können im Runtime Modus über die Bewegungstasten angesprungen werden. Man unterscheidet zwischen:

- Alphacursorobjekten (EA-Objekte) und
- Schaltcursorobjekten.

Ein-/Ausgabe-Objekte werden über den Alphacursor (Tabulator-Taste bzw. UMSCHALT Tabulator-Taste) angewählt.

Alle Bedienelemente (mit Maus-, Tastatur- oder beliebiger Bedienung) können in die Bedienung über den Schaltcursor einbezogen werden. Die Ein-/ Ausgabefelder können sowohl in die Alphacursor- als auch die Schaltcursor-Bedienung integriert werden.

TAB-Reihenfolge

Durch die sogenannte *TAB-Reihenfolge* (über Menüpunkt *Bearbeiten* → *TAB-Reihenfolge* → *Alphacursor* oder → *Schaltcursor* einstellbar) haben Sie Einfluß darauf, in welcher Reihenfolge bedienbare Objekte im Runtime Modus angesprungen werden. Das aktuell angewählte Objekt kann im Runtime visualisiert werden. Dies ist der Runtime-Cursor, der auch abgeschaltet werden kann (*Rechnereigenschaften* → *Graphics Runtime*). Buttons, welche im Windows-Stil dargestellt werden, werden bei Anwahl immer mit einem gestrichelten Viereck in der Taste gezeigt.

Das Bewegen über die bedienbaren Elemente ist abhängig von den Einstellungen für das *Graphics Runtime* (*Rechnereigenschaften* → *Graphics Runtime*).

Bewegung	Standard-Tasten	Tasteneinstellungen
Auf, Ab links, rechts	Pfeiltasten oder Tabulator (nächstes) oder UMSCHALT Tabulator (vorheriges)	weitere Tasteneinstellungen über <i>Rechnereigenschaften</i> → <i>Graphics Runtime</i> → <i>Cursorsteuerung-Tasten</i>
Alpha-/Schalt- cursor	Schaltcursor	Umschalten zwischen Alpha- und Schaltcursor über Hotkeys (<i>Rechnereigen- schaften</i> → <i>Graphics Runtime</i> → <i>Hotkeys</i>) oder eigene Tasten (mit Verwendung der internen Funktion <i>SetCursorMode</i>)
Tabellen- bewegung (Cursorgruppe)	Normal, d.h. jeweils zeilenweise Bearbeitung wenn der Cursor das Ende der Cursorgruppe erreicht hat, bleibt er an dieser Position stehen.	Änderung des Verhaltens über <i>Rechnereigenschaften</i> → <i>Graphics Runtime</i> → <i>Cursorsteuerung-Verhalten</i>

Ein-/Ausgabefelder

Projektierte Ein-/Ausgabefelder sind nach der Auswahl über die Tastatur direkt bedienbar, d.h. Sie können sofort Ihre neuen Daten eingeben. Ein reines Ausgabefeld (*Eigenschaften → Ausgabe/Eingabe → Feldtyp → Ausgabe*) kann nicht bedient werden.

Das Bestätigen der Eingabe erfolgt abhängig von der projektierten Eigenschaft (*Eigenschaften → Ausgabe/Eingabe → Übernahme bei Verlassen*) über die EINGABE-Taste. Die Abbruchtaste (ESC) dagegen beendet die Eingabe, ohne daß eine Änderung des Wertes vorgenommen wird.

Weitere Eingabeobjekte

Neben den typischen Analog-Eingabefeldern sind unter den Windows-Anwendungen noch weitere Eingabemöglichkeiten bekannt. Diese speziellen Objekte finden Sie in der Objektpalette unter *Windows-Objekte*

- Checkbox
- Radiobox

Das Setzen der einzelnen Auswahlfelder der Check- oder Radiobox erfolgt über die Leertaste und das Bewegen über die einzelnen Komponenten in der Box mittels der Auf-/Ab-Tasten (z.B. Richtungstasten). Dies ist die bereits standardmäßig hinterlegte Tastenbelegung.

Ein weiteres Eingabeobjekt stellt das **Textlistenobjekt** dar. Über eine, abhängig von den projektierten Einträgen, aufgeschlagene Liste kann eine Auswahl getroffen werden:

Die Bedienung kann ebenfalls über die Standard-Tastatur erfolgen. Es muß keine spezielle Projektierung für die Tastaturbedienung hinterlegt werden.

Das Aufschlagen der Liste erfolgt über die EINGABE-Taste, das Bewegen in der Liste über die Auf-/Ab-Tasten und das Bestätigen der aktuellen Auswahl über die EINGABE-Taste.

Weitere Eingabe-Objekte könnten über die OCX-Elemente in WinCC genutzt werden. Deren Bedienung und Projektierung ist jedoch abhängig von den verfügbaren Ereignissen und Eigenschaften, die objektbezogen definiert sind. Dies muß im Einzelfall geklärt werden.

3.3.10.3 Alarm Logging Funktionstasten zu den Toolbar-Tasten

In den Meldefenstern werden unterschiedliche Bedientasten in der Toolbar eingestellt, die standardmäßig über die Maus bedient werden.

Die häufigsten Bedienungen in einem Meldefenster sind

- Anwahl einer Meldung für die Quittierung
- Auf-/Abwärts-Bewegen in der Meldeliste
- Blättern in der Meldeliste

Hinweis:

Beim Aufschlagen des Meldefensters bzw. durch eine weitere Bedienung muß die Bedienbarkeit im Meldefenster (Applikationsfenster) und nicht im Hauptfenster liegen. Abhängig von der aktuellen Bedienbarkeit wirken die Tastenbedienungen (bzw. Funktionstasten) auf die Funktionstastenleiste des Hauptfensters oder auf die hinterlegten Tastenbedienungen des Meldefensters.



Dies kann zum Beispiel über das Setzen des aktuellen Bedienfokus in diesen Fensterbereich erreicht werden. Der Bedienfokus wird normalerweise über die Maus durch das Anklicken gesetzt.

Über die Tastatur kann der Fokus über folgende projektierbaren Wege in Meldefenster gesetzt werden:

- Wechseln des Fensters über Hotkey
- Setzen des Fokus über eine Bedientaste oder
- direktes Setzen des Fokus auf ein definiertes Element im Meldefenster bei Bildanwahl.

Das Wechseln in das Meldefenster mittels einer Schnellbedienung (Hotkey), die für alle Fensterwechsel, d.h. auch bei Kurvenfenstern z.B. in gleicher Art eingesetzt werden kann, wird in den Anlauf-Parametern des Graphics Runtime festgelegt. Unter

Rechnereigenschaften → *Graphics Runtime* → *Hotkeys* → *Fenster umschalten* wird die Tastenkombination (z.B. STRG W) eingetragen.

Nach dem Anwählen des Meldefensters kann über diese Tastenfolge direkt die Bedienung der Tasten der Toolbar erfolgen.

Das direkte Setzen des Bedienfokus auf das Meldefenster dagegen wird mittels der Internen Funktion *Set_Focus* realisiert. Eine *C-Aktion* zu einer Tastenbedienung oder bei Bildanwahl (*Bild-Objekt* → *Ereignis* → *Sonstige* → *Bildanwahl*) kann daher das Aktivieren der Bedienung des Meldefensters beeinflussen.

Die Funktionen für den Bildfokus finden Sie unter *Interne Funktionen* → *graphics* → *Set* → *focus*. Zum Beispiel wird für das Setzen des Bedienfokus folgende Funktion aufgerufen:

```
Set_Focus(Bildname, Objektname);
```

Für die Parameter muß der Name des Hauptfensters (Bildname) sowie der des Applikationsfensters (Objektname) eingetragen werden.

Die Anwahl einer Meldung im Meldefenster erfolgt über die Anwahl der Meldezeile. Bei der Anwahl des Meldefensters steht der aktuelle Cursor auf der jüngsten Meldung (letzte Meldung im Meldebild). Ob in dem Fenster eine Meldung angewählt werden kann bzw. ob das Blättern in den Meldungen möglich ist, hängt von der Aktivschaltung des Scroll-Mechanismus ab.

Das Ein-/Ausschalten des Bild-Scrollens kann über eine eigene Taste in der Toolbar umgeschaltet werden oder direkt in der Projektierung des Meldfenster-Templates eingestellt werden. Das aktive Einschalten des Scroll-Mechanismus bei Bildaufschlag erfolgt im Editor *Alarm Logging* → *Meldfenster-Vorlagen* → *Parameter* → *AutoScrolling*.



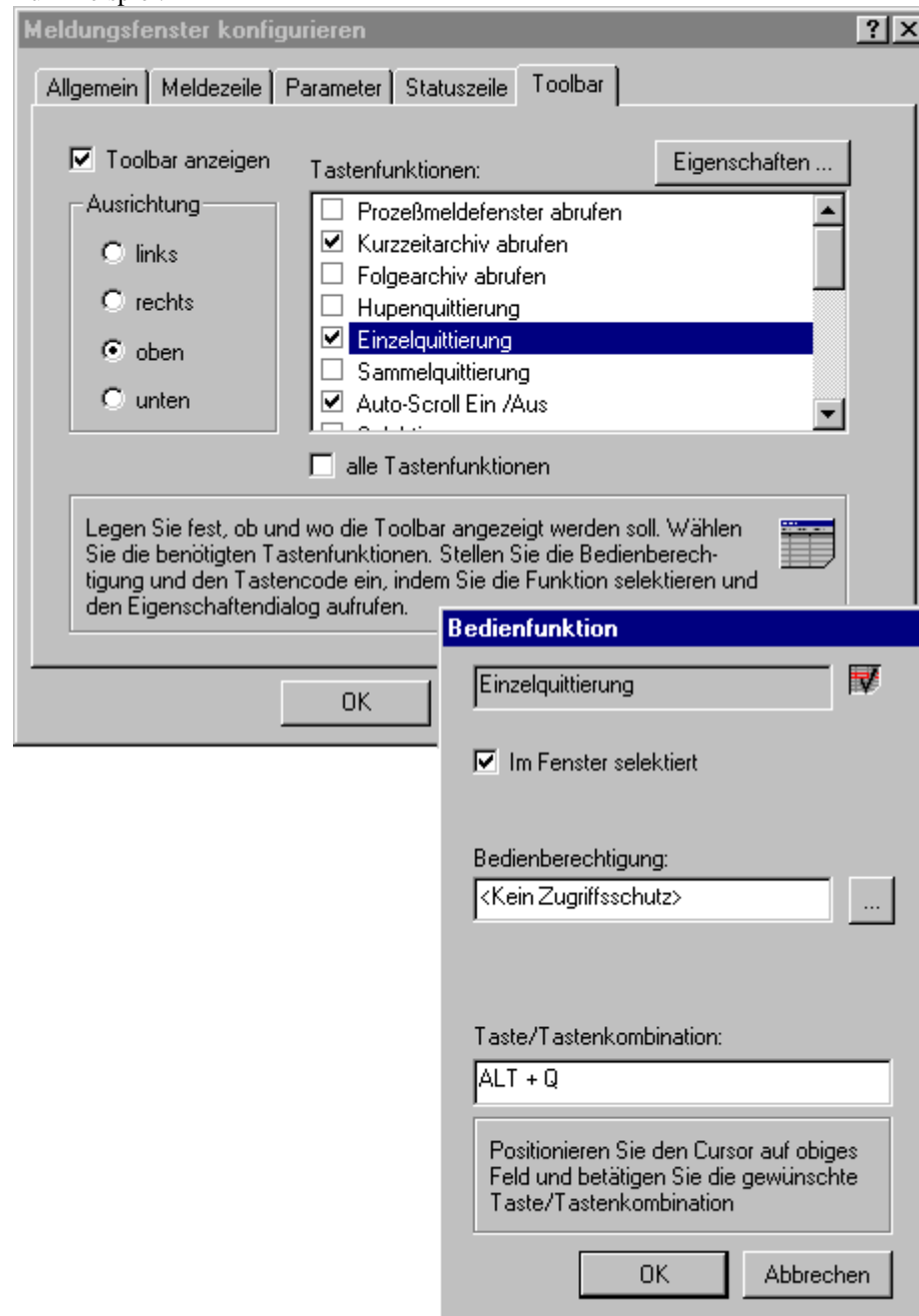
Ist das Scrollen sowie der Bedienfokus in dem Meldfenster aktiv, können die Bewegungen folgendermaßen ausgeführt werden:

Bewegung	Standard-Tasten	Tasteneinstellungen
Auf, Ab in der Meldeleiste	Richtungstasten	einzelne Meldezeilen
Anfang, Ende in der Meldeliste	Pos1-, Ende-Taste	Anfang bzw. Ende des Meldebildes
Blättern	Blättertasten	mehrere Meldezeilen

Um die Bedientasten der Toolbar wie z.B. die Quittierung der angewählten Meldung zu aktivieren, kann neben der Standard-Mausbedienung auch eine Funktionstastenbedienung definiert werden.

Für jede in der Toolbar eingeblendete Taste (*Alarm Logging* → *Meldfenster-Vorlage* → *Toolbar* → *Taste angekreuzt und angewählt* → *Eigenschaften*) kann eine zugehörige Tastenbedienung in den Eigenschaften hinterlegt werden.

Zum Beispiel:



Über diese Projektierungsschritte können alle eingesetzten Bedientasten für das Meldefenster mit einer Tastenkombination versehen werden. Eine Tastenbedienung im Meldefenster ist daher zusätzlich zu definieren.

3.3.10.4 Alarm Logging anlagenspezifisch entworfene Toolbar-Tasten

Die gesamten Toolbar-Tasten sind von WinCC vorgegeben und können nicht verändert werden. Wird für die zu projektierende Anlage ein vorgegebenes Layout der Tasten vorgeschrieben, so muß die WinCC Toolbar deaktiviert werden (d.h. keine Toolbar) und die zugehörigen Tasten selbst entworfen werden. Alle diese neuen Tastenobjekte können nach den Wunschvorstellungen des Kunden z.B. mit Bildern versehen werden.

Die Funktionalität zu den einzelnen Tasten muß aber noch als zugehörige Aktion projektiert werden. In der *C-Aktion* des zugehörigen Ereignisses (z.B. Taste drücken) muß die entsprechende *Standard-Funktion* aus dem Funktionsbaum ausgewählt werden.

Die für die Tastenbedienung zur Verfügung gestellten Funktionen finden Sie unter *Standard-Funktionen* → *alarm*. Für jede Taste aus der Toolbar findet sich eine korrespondierende Funktion in der Liste wieder. Zum Beispiel wird für die Quittiertaste folgende Funktion aufgerufen:

```
OnBtnSinglAckn(Fenstername);
```

Für den Parameter muß der Name der *Meldefenster Vorlage* eingetragen werden. Diese Aktionen können auch für selbstentworfene Bedientasten über Mausbedienung genutzt werden. Einige derartige Tastenbeispiele finden Sie in den Optionspaketen zum Alarmsystem (z.B. *Basic Process Control* - Hupenquittierung etc.).

3.3.10.5 Tag Logging Funktionstasten zu den Toolbar-Tasten

In den Kurven- bzw. Tabellenfenstern zur Darstellung von Meßwerten - auch Trendanzeigen genannt - werden unterschiedliche Bedientasten in der Toolbar eingestellt, die standardmäßig über die Maus bedient werden.

Die häufigsten Bedienungen in einem Kurvenfenster sind

- Blättern in den Meßwerten (Zeitachse)
- Zeitbereich auswählen
- Kurven auswählen
- Leselineal bedienen

Nach dem Aufschlagen des Kurvenfensters wird abhängig von der Projektierung der aktuelle Kurvenverlauf angezeigt.

Hinweis:

Beim Aufschlagen des Kurvenfensters muß die Bedienbarkeit im Kurvenfenster (Applikationsfenster) und nicht im Hauptfenster liegen. Abhängig von der aktuellen Bedienbarkeit wirken die Tastenbedienungen (bzw. Funktionstasten) auf die *Funktionstastenleiste* des Hauptfensters oder auf die hinterlegten Tastenbedienungen des Kurven- bzw. Tabellenfensters.

Dies kann zum Beispiel über das Setzen des aktuellen Bedienfokus in diesen Fensterbereich erreicht werden. Der Bedienfokus wird normalerweise über die Maus durch das Anklicken gesetzt.

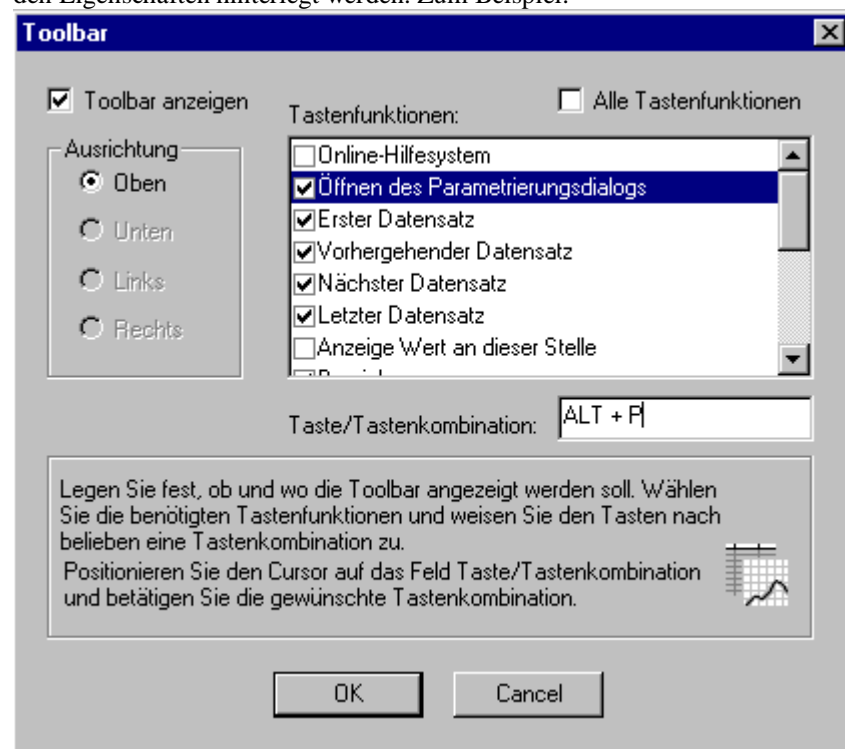
Über die Tastatur kann der Fokus über folgende, projektierbaren Wege in das Kurven- bzw. Tabellenfenster gesetzt werden:

- Wechseln des Fensters über Hotkey
- Setzen des Fokus über eine Bedientaste oder
- direktes Setzen des Fokus auf ein definiertes Element im Meldefenster bei Bildanwahl.

Die Realisierung dieser verschiedenen Varianten finden Sie in diesem Kapitel unter der Beschreibung des Alarm Logging.

Um die Bedientasten der Toolbar wie z.B. Auswahl eines Zeitbereiches zu aktivieren, kann neben der Standard-Mausbedienung auch eine Funktionstastenbedienung definiert werden. Standardmäßig sind die einzelnen Tasten mit den Funktionstasten F1 bis F10 belegt.

Für jede in der Toolbar eingeblendete Taste (*Tag Logging* → *Kurvenfenstervorlage* → *Template* → *Toolbar* → *Konfigurieren...*) kann eine eigene zugehörige Tastenbedienung in den Eigenschaften hinterlegt werden. Zum Beispiel:



Über diese Projektierungsschritte können alle eingesetzten Bedientasten für das Kurven- oder Tabellenfenster mit einer Tastenkombination versehen werden. Eine Tastenbedienung für die Kurven- bzw. Tabellenfenster ist daher zusätzlich zu definieren.

In dem Kurvenfenster können folgende Standardtasten nach dem Aktivieren der jeweiligen Funktionstaste eingesetzt werden:

Bewegung	Standard-Tasten	Tasteneinstellungen
Leselineal	Richtungstasten	Leselinie nach links bzw. rechts bewegen
Zoomen	Zoomausschnitt wählen	Ersatz-Eingabehilfe für Maus einstellen und aktivieren (siehe Systemeinstellungen) EINFG und Richtungstasten ermöglichen die Festlegung des Zoomausschnittes
Dialoge z.B. Archivvariablenauswahl	Tabulatortaste	zur Bewegung zwischen den Eingabefeldern
	Richtungstasten	innerhalb der Variablenauswahl bzw. Registerauswahl bewegen
	+ Taste (- Taste)	Aufblenden bzw. Schließen des Baumes der Variablen des Archives
	Leertaste	Auswählen bzw. Auswahl zurücknehmen
	EINGABE-Taste	Dialogbox bestätigen und beenden
	ESC-Taste	Dialogbox abrechnen.

Tag Logging anlagenspezifisch entworfene Toolbar-Tasten

Die gesamten Toolbar-Tasten sind von WinCC vorgegeben und können in der Gestalt nicht verändert werden. Wird für die zu projektierende Anlage ein vorgegebenes Layout der Tasten vorgeschrieben, so muß die WinCC Toolbar abgeblendet werden und die zugehörigen Tasten selbst entworfen werden. Alle diese neuen Tastenobjekte können nach den Wunschvorstellungen des Kunden z.B. mit Bildern versehen werden. Die Funktionalität zu den einzelnen Tasten muß aber noch als zugehörige Aktion projektiert werden. In der *C-Aktion* des zugehörigen Ereignisses (z.B. Taste drücken) muß die entsprechende *Standard-Funktion* aus dem Funktionsbaum ausgewählt werden.

Die für die Tastenbedienung zur Verfügung gestellten Funktionen finden Sie unter *Standard-Funktionen* → *TAGLOG* (oder zusätzlich → *TEMPLATE* für die Tabellenfunktionen). Für jede Taste aus der Toolbar findet sich eine korrespondierende Funktion in der Liste wieder. Zum Beispiel wird das Leselineal über folgende Funktion aufgerufen:

```
TlgTrendWindowPressLinealButton(Fenstername) ;
```

Für den Parameter muß der Name der *Kurvenfenstervorlage* eingetragen werden.

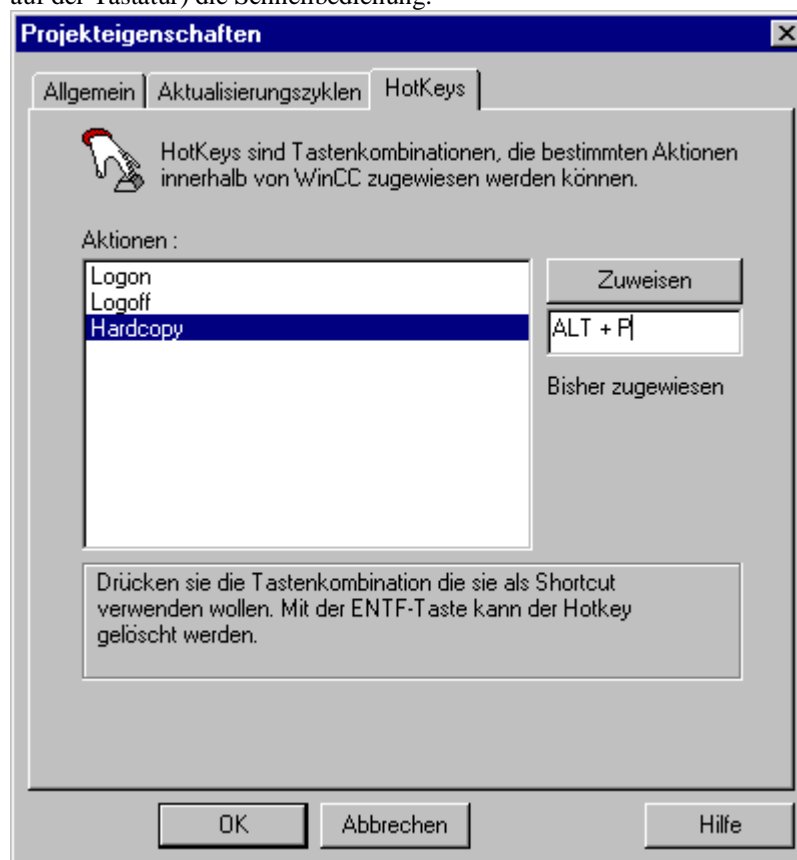
Diese Aktionen können auch für selbstentworfene Bedientasten über Mausbedienung genutzt werden.

3.3.10.6 Druckauftrag anstoßen

Ein Druckauftrag kann über mehrere Wege angestoßen werden. Im *Control Center* zum Beispiel wird direkt über die Auswahl in der Liste der Druckaufträge der Druck aktiviert. Im Anlagenbild selbst kann aber auch eine Taste kreiert werden, um den Druckauftrag über die Taste anzustoßen.

Für die Meldelisten existiert diese Taste bereits in der Toolbar und kann, wie auf den vorherigen Seiten beschrieben, mit einer Funktionstaste oder einer eigens entworfenen Taste aktiviert werden.

Ein Bildschirmabzug - die sogenannte Hardcopy - ist mittels einer Kurzbedienung in jedem Bild aktivierbar. Dieser Hotkey wird in Projekteigenschaften global eingestellt. Rufen Sie dazu im *Control Center* die *Projekteigenschaften* → *Hotkey* → *Hardcopy* auf und definieren Sie durch direkte Eingabe der Tastenkombination (Drücken der jeweiligen Taste auf der Tastatur) die Schnellbedienung.



Wird eine eigens entworfene Taste für den Druck eines definierten Druckauftrages im Anlagenbild projiziert, so muß der Anstoß über eine *C-Aktion* erfolgen. Die Taste wird wie am Anfang des Kapitels beschrieben, z.B. über eine Funktionstaste (siehe Hotkey) oder über die Tastaturbedienung (z.B. Taste D) bedient. Die *C-Aktion* muß dementsprechend an dem zugehörigen Ereignis (z.B. *Mausklick* oder *Tastatur - drücken*) projiziert werden. Von WinCC wird diese Funktionalität mit den Funktionen in *Standard-Funktionen* → *Report* → *ReportJob* zur Verfügung gestellt.

```
ReportJob("Documentation Control Center", "PRINTJOB");
```

Die Funktion erhält den Namen des Druckauftrages sowie als Methodennamen das Drucken auf Drucker bzw. als Vorschau.

3.3.10.7 An- oder Abmelden

Neben den einstellbaren Hotkeys für den Login- bzw. Logout-Vorgang kann auch eine Taste projiziert werden, die das Aufblenden der Login-Dialogbox bewirkt. Das Abmelden über eine Tastenbedienung ist ebenfalls möglich. Hierzu muß jeweils eine eigene Taste entworfen werden, die zum Beispiel neben dem Mausclick auch über die Tastatur bedienbar ist. Eine Funktionstastenbedienung ist ebenfalls über die Hotkey-Eigenschaft der Taste einstellbar. Die verschiedenen Varianten für eine Tastenbedienung werden am Anfang des Kapitels im Detail beschrieben. Die Funktion, die für das An- bzw. Abmelden eingesetzt werden muß, ist eine WinCC Applikationsfunktion. Der Einsatz dieser Funktion muß als *C-Aktion* projiziert werden. Legen Sie die *C-Aktion* zum Beispiel an das Ereignis *Mausklick* oder das Ereignis *Taste drücken*.

Für das Abmelden wird folgende Funktion eingesetzt:

```
#pragma code ("useadmin.dll")
#include "PWRT_api.h"
#pragma code()
PWRTLogout();
```

Das Anmelden erfolgt mit der WinCC-C-Applikationsfunktion *PWRTLogin()*. Ein Beispiel für den Einsatz dieser Funktion wäre:

```
#pragma code ("useadmin.dll")
#include "PWRT_api.h"
#pragma code()
PWRTLogin(c);
```

Die eingeblendete Dialogbox kann mit den Standard-Tasten bedient werden:

Bewegung	Standard-Tasten	Tasteneinstellungen
einzelne Eingabefelder	Tabulator-Taste (vorwärts) oder UMSCHALT und Tabulator-Taste (rückwärts) Richtungstasten	Leselinie nach links bzw. rechts bewegen
Bestätigen (OK)	EINGABE-Taste	Dialogbox beenden und Eingabe bestätigen
Abbrechen	ESC-Taste	Dialogbox bzw. Eingabe abbrechen

3.3.11 Bildbausteintechnik

Die Bausteintechnik ist ein wesentlicher Ansatz für die schnelle und einfache Projektierung sowie die Wiederverwendbarkeit und Wartbarkeit von projektierten Bildkomponenten.

Eine projektierte Prozeßbox wird zum Beispiel für mehrere gleichartige Prozeßkomponenten (z.B. Ventile oder Regler) eingesetzt.

Das einmalig, projektierte Bildfenster kann nun für die im Projekt zu bedienenden und zu visualisierenden Regelbausteine nach folgenden Prinzipien weitergenutzt werden:

- Kopieren eines Bildfensters und jeweilige Neuverbindung der Variablenfelder
- Einsatz eines Bildfensters, dessen Variablenfelder beim Aufruf zugewiesen werden (indirekte Verbindung)
- Einsatz von Anwenderobjekten mit Prototypen und daraus erzeugten Objekten
- Erstellung von Prototypbildern und deren Einbindung
- Erstellung von OCX-Bildbausteinen und Einbinden als WinCC-OCX-Objekte

Vergleich der verschiedenen Techniken

Diese Techniken sind in ihrem Einsatz, der Komplexität der Projektierung und ihren Möglichkeiten sehr unterschiedlich, so daß wir zunächst einen Vergleich der Alternativen vornehmen.

Art	Vorteil	Nachteil
Kopie von Bildfenstern	einfache Vorgehensweise	alle Objektverbindungen müssen geändert werden Änderungen im Bildaufbau führen zu aufwendigen Nachbearbeitungen
Bildfenster mit indirekter Verbindung	<ul style="list-style-type: none"> • einmaliger Aufbau des Bildfensters mit einfachen <i>C-Aktionen</i> • Wiederverwendung ohne Kopie des Basis-Bildfensters 	Änderungen im Bildaufbau führen zu aufwendigen Nachbearbeitungen
Anwenderobjekte	einmaliger Aufbau des Objektes mit Verbindung über vorhandene Dynamic-Wizards	<ul style="list-style-type: none"> • Änderungen im Bildaufbau führen zu Nachbearbeitungen, d.h. neue Bilderzeugung • nicht zentral änderbar
Prototypbilder	<ul style="list-style-type: none"> • einmaliger Aufbau des Objektes • zentral änderbar 	(gute) C-Kenntnisse nötig
OCX	<ul style="list-style-type: none"> • einfaches Einbinden in die Projektierung von WinCC als Objekt im Bild • eine nachträgliche Änderung des OCX-Objektes führt zu keiner Nachbearbeitung in den 	muß per Programmierung erstellt werden (C++, VB 5), kann nicht mittels WinCC Projektierung erstellt werden.

Art	Vorteil	Nachteil
	erzeugten Objekten, außer bei Änderungen der Objekt-Eigenschaften <ul style="list-style-type: none"> • sehr performant • weitere grafische Möglichkeiten • Zukauf von neuen Objekten (z.B. PCS7-Bausteine) 	

Werden im Projekt nur wenige, einfache Bildbausteine eingesetzt, so genügt dem Projektierer eine der ersten Varianten. Diese können ohne größere Einarbeitung umgesetzt werden.

Für einfache Objekte mit kleiner bis mittlerer Komplexität und Variablenanbindung eignet sich das Anwenderobjekt sehr gut. Bedarf dieses Objekt mehrerer vorhersehbarer Änderungen, so lohnt sich eine Einarbeitung in das Konzept der Prototypbilder. Sind die grafischen Bausteine von komplexer Natur bzw. wird eine umfangreichere Verarbeitungsleistung benötigt, so ist die OCX-Technik zu bevorzugen. Die verfügbaren OCX-Objekte werden branchenbezogen in Zukunft immer stärker wachsen.

Im folgenden zeigen wir die verschiedenen Arten der Projektierung von Bildbausteinen und deren Verwendung in den Anlagenbildern. Sie können sich auf diesem Wege selbst ein Bild von den unterschiedlichen Varianten und deren Einsatzgebieten in Ihren Projekten machen.

3.3.11.1 Prozeßbox als Bildbaustein

Zur Darstellung der aktuellen Zustände eines Aggregates (Regler, Ventile, Motoren etc.) bzw. zur Vorgabe von Sollwerten werden in den Anlagenbildern spezifische Informationsboxen eingeblendet. Diese Prozeßboxen beinhalten typischerweise sowohl aktuelle Zustände (Istwerte) als auch Vorgabewerte, die über den privilegierten Bediener eingegeben werden können:

Erstellen der Informationsbox

Diese Informationsbox wird als Bildfenster kreiert, deren Komponenten mit den zugehörigen (Prozeß-)Variablen verbunden werden.

Schritt	Typ	Projektierung
1	Datenstrukturen	Definieren Sie die im Bildbaustein einzusetzenden Datenstrukturen über den Variablenhaushalt, z.B. Motor mit Istwert, Sollwert, Ein-/ Ausschalter:
2	Bildbaustein	Projektieren Sie über den <i>Graphics Designer</i> ein Bild, das die Aggregatzustände zeigt, z.B. Balken und E/A-Felder sowie Bedientasten. Die Größe des Bildfensters (Eigenschaft Bild-Objekt - X-Größe und Y-Größe) muß der Zielgröße des Bildfensters entsprechen.
3	Variablen definieren	Definieren Sie die (Prozeß-)Variablen im Variablenhaushalt. z.B. Motor_T01 vom (Struktur-)Datentyp Motortyp, der für die Prozeßbox verwendet wird.

Schritt	Typ	Projektierung
4	Variablen- verbindung	Dynamisieren Sie nun die einzelnen Bildkomponenten z.B. EA-Felder, Balken etc. durch Verbindung mit den zugehörigen (Prozeß-)Variablen
5	Bildfenster	In dem Anlagenbild legen Sie ein Bildfenster-Objekt an und verbinden dieses mit dem unter Schritt 2-4 entworfenen Bildfensterinhalt über die Eigenschaft Bildfenstername.
6	Eigenschaften Einstellungen	Dieses Bildfenster-Objekt soll bei Bildaufschlag noch nicht angezeigt werden. Deshalb muß die Eigenschaft <i>Anzeige</i> statisch auf nein gesetzt werden. Das Aussehen des Bildfensters mit den Windows-Tasten sowie Titel etc. müssen in den Eigenschaften des Bildfensters ebenfalls noch festgelegt werden.
7	Aufruf des Bildfensters	Dieses Bildfenster muß z.B. über eine Bedientaste oder über die Bedienung des Aggregates selbst aufgeblendet werden. Entwerfen Sie eine Bedientaste, die mit dem Einblenden des Bildfenster-Objektes verbunden wird (z.B. über <i>Direktverbindung</i>)

Dieses Bildfensterobjekt, der Bildfensterinhalt sowie der zugehörige Aufruf des Bildfensters (Bedientaste) können bei weiteren Aggregaten in ähnlicher Form wieder eingesetzt werden. Dazu muß jeweils das Bildfenster-Objekt, der Bildbaustein sowie die Bedientaste kopiert werden. Die Referenzen müssen jedesmal angepaßt werden. Für den Kopiervorgang können das Bildfenster-Objekt sowie die Bedientaste mittels Drag&Drop in die Grafikbibliothek (z.B. Projekt-Bibliothek) gelegt werden.

Anpassen der Bildbausteine

Es müssen daher folgende, einzelne Schritte bei der Verwendung des erstellten Bildbausteins durchgeführt werden:

Schritt	Typ	Projektierung
1	Prozeß- variablen	Definieren Sie eine neue Prozeßvariable, z.B. Motor_T02 zu der definierten Datenstruktur
2	Kopie des Bildbausteins	Legen Sie eine Kopie des Bildfensterinhaltes an (Motort02.PDL) und ändern Sie alle fest hinterlegten Referenzen (z.B. statt Motor_T01.Istwert nun Motor_T02.Istwert).
3	Kopie des Bildfensters	Legen Sie eine Kopie des Bildfenster-Objektes im Ziel-Anlagenbild an (mittels Drag&Drop aus der Grafikbibliothek). Passen Sie die Referenz zum Bildfensterinhalt unter der <i>Eigenschaft</i> → <i>Bildname</i> an (Motor02.PDL).
4	Kopie der Bedientaste	Legen Sie eine Kopie der Bedientaste im Ziel-Anlagenbild an (mittels Drag&Drop aus der Grafikbibliothek). Passen Sie die Referenz zu dem neuen Bildfenster-Objekt in der <i>Direktverbindung</i> (<i>Objekt</i> → <i>Bildfenster2</i> → <i>Anzeige</i>) an.

Auf diese Weise können die einzelnen Bildfenster und deren Inhalt pro Aggregat erstellt und über Kopiervorgänge wieder verwendet werden. Wie bereits sichtbar, besteht der Aufwand jeweils in der Anpassung der fest hinterlegten Referenzen bei dem Bildfensterinhalt zum Beispiel. Aus diesem Grunde kann über den Weg der indirekten

Adressierung eine einfachere Wiederverwendbarkeit erreicht werden. Der Anpassungsaufwand soll auf ein Minimum reduziert werden.

Als alternative Lösung kann der Bildbaustein auch ohne Bildfensteranbindung projektiert werden. Dies bedeutet, daß der Bildbaustein selbst als nicht angezeigtes Objekt im Anlagenbild projektiert wird. Dies hat aber bei der Änderung des Bildbausteines den erheblichen Nachteil, daß in allen Bildern, in denen dieser Bildbaustein verwendet wird, jeweils eine Änderung durchgeführt werden muß.

3.3.11.2 Bildbaustein mit indirekter Adressierung

Bisher wurden die einzelnen Komponenten des Bildbausteins fest mit den zugehörigen (Prozeß-) Variablen verbunden. Wird die Verbindung nicht durch eine fixe Projektierung, sondern dynamisch zur Laufzeit ermittelt, kann der entworfene Bildbaustein wesentlich flexibler eingesetzt werden. Diese dynamische Verbindung von (Prozeß-) Variablen wird über die indirekte Adressierung der einzelnen Komponenten im Bildbaustein realisiert. Dies bedeutet, daß nicht direkt mit der (Prozeß-) Variablen verbunden wird, sondern nur mit dem *Container*, der den aktuellen Namen der zugehörigen (Prozeß-) Variablen zur Laufzeit tragen wird.

Der Anpassungs- sowie Wiederverwendbarkeitscharakter eines Bildbausteins kann damit wesentlich vereinfacht werden.

Die Projektierung wird ähnlich der Schritte wie zuvor beschrieben durchgeführt. Die Schritte im einzelnen:

Schritt	Typ	Projektierung
1	Datenfestlegung	Definieren der im Bildbaustein einzusetzenden Daten über den Variablenhaushalt, z.B. Motor001_Istwert, Motor001_Sollwert, Motor001_Schalter einerseits und Festlegung der Namens-Container für die einzelnen Komponenten, die im Bildbaustein verwendet werden sollen, z.B. Istw_Name, Sollw_Name, etc. andererseits. Diese Variablen initialisieren Sie mit einem Namen, z.B. Motor001_Sollwert.
2	Bildbaustein	Projektieren Sie über den <i>Graphics Designer</i> ein Bild, das die Aggregatzustände zeigt, z.B. Balken und E/A-Felder sowie Bedientasten. Die Größe des Bildfensters (Eigenschaft Bild-Objekt - X-Größe und Y-Größe) muß der Zielgröße des Bildfensters entsprechen.
3	Variablenverbindung	Dynamisieren Sie nun die einzelnen Bildkomponenten z.B. EA-Felder, Balken etc. mit Verbindung der zugehörigen Container-Variablen, die den Namen der entsprechenden Variablen enthalten. Bei der Verbindung muß aber nun hinterlegt werden, daß die Variable nur der Name der eigentlichen (Prozeß-) Variablen ist. Deshalb muß jeweils in der Spalte <i>indirek.Adr.</i> angekreuzt werden.
4	Bildfenster	In dem Anlagenbild legen Sie ein Bildfenster-Objekt an und verbinden dieses mit dem unter Schritt 2-3 entworfenen Bildfensterinhalt über die Eigenschaft Bildfenstername.
5	Eigenschaften-Einstellungen	Dieses Bildfenster-Objekt soll bei Bildaufschlag noch nicht angezeigt werden. Deshalb muß die Eigenschaft <i>Anzeige</i> statisch auf nein gesetzt werden. Das Aussehen des Bildfensters mit den Windows-Tasten sowie

Schritt	Typ	Projektierung
		Titel etc. müssen in den Eigenschaften des Bildfensters ebenfalls noch festgelegt werden.
6	Aufruf des Bildfensters	Dieses Bildfenster muß z.B. über eine Bedientaste oder über die Bedienung des Aggregates selbst aufgeblendet werden. Entwerfen Sie eine Bedientaste, die mit dem Einblenden des Bildfenster-Objektes verbunden wird (z.B. über <i>Direktverbindung</i>).
7	Grafikbibliothek	Das Bildfenster-Objekt sowie die Bedientaste werden mittels Drag&Drop zur Wiederverwendung in die Bibliothek gelegt.

3.3.11.3 Anwenderobjekte

Mittels Anwenderobjekten und den zugehörigen Dynamic-Wizards können Bildbausteine erstellt werden, die einfach wiederverwendet werden können. Die erstellte Kopie des Bildbausteines kann mit einer einfachen Wizard-Projektierung mit den zugehörigen, aktuellen (Prozeß-)Variablen verbunden werden.

Ein Anwenderobjekt ist ein vom Projektierer entworfenes grafisches Objekt (z.B. Kombination mehrerer), dessen viele Eigenschaften und Ereignisse mittels eines Konfigurationsdialoges auf die wesentlichen Eigenschaften und Eigenschaften beschränkt werden. Dieses Anwenderobjekt wird mittels des zugehörigen Wizard als Prototyp dynamisiert.

Im einzelnen sind folgende Schritte notwendig:

Schritt	Typ	Projektierung
1	Datenstrukturen	Definieren der im Bildbaustein einzusetzenden Datenstrukturen über den Variablenhaushalt.
2	Bildbaustein	Projektieren Sie über den <i>Graphics Designer</i> ein Anwenderobjekt mit den benutzerdefinierten Eigenschaften

Ein Anwenderobjekt wird aus einer Gruppe von WinCC-Objekten gebildet. An diesen Objekten ist zunächst keine Dynamik projektiert. Alle Objekte, die zu dem Anwenderobjekt zusammengefaßt werden sollen, werden selektiert und der Konfigurationsdialog des Anwenderobjektes wird aufgerufen:

In diesem Dialog werden nun alle Eigenschaften der Objekte zur Eigenschaft des Anwenderobjektes erklärt, die später dynamisiert werden sollen. Die Basis-Eigenschaften für ein Objekt (z.B. Position und Größe) wurden bereits für das Anwenderobjekt hinterlegt. Jede einzelne Eigenschaft der zusammengefaßten Objekte kann im Dialog ausgewählt werden und per Drag&Drop als *Benutzerdefinierte Eigenschaft* bzw. *Ereignis* dem neuen Anwenderobjekt hinzugefügt werden.

Jeder dieser Eigenschaften kann ein neuer (sprachenunabhängiger) Attributname vom Anwender vergeben werden sowie auch der sprachenabhängige Eigenschaftsname (z.B. bei englischer Projektierung). Eigenschaften, die im Eigenschaftendialog nicht sichtbar dargestellt werden sollen, aber z.B. bei Scripten genutzt werden, können mittels des Zeichens @ verborgen werden. Damit können nur wenige (zu dynamisierenden) Eigenschaften und Ereignisse nach außen geführt werden. Alle anderen werden verborgen.

Das entworfene Anwenderobjekt muß nun dynamisiert werden. Dazu wird ein Wizard zur Verfügung gestellt:

Schritt	Typ	Projektierung
3	Dynamisieren	<p>Rufen Sie den Dynamic-Wizard <i>Prototyp dynamisieren</i> auf.</p> <p>Verbinden Sie schablonenmäßig (d.h. prototypisch) jede einzelne Eigenschaft des Objektes mit der zugehörigen Strukturkomponente der definierten Datenstruktur.</p> <p>Über den Variablen-Browser wird der <i>Strukturmember</i> zur Verschaltung ausgewählt.</p> <p>Der Wizard speichert aber nur den Namen der Strukturkomponente an der verbundenen Eigenschaft (z.B. .Value). Es muß jede einzelne Eigenschaft separat verbunden werden.</p> <p>Dieses Objekt ist nun ein dynamisiertes Objekt, das aber nur prototypisch verbunden wurde und zur Laufzeit noch nicht lebt. Es kann daher zur Laufzeit nicht aktualisiert werden.</p>
4	Ablegen in die Grafikbibliothek	Legen Sie diesen Prototyp als Objekt in die Grafikbibliothek

Das Prototyp-Anwenderobjekt wird z.B. für die mehrfache Wiederverwendung in die Grafikbibliothek gelegt. Ein Beispiel für ein dynamisches Objekt sind die Zeigerinstrumente in der WinCC-Bibliothek (Anwenderbibliothek, Anwenderobjekte, Zeigerinstrumente).

Das Prototyp-Objekt wird in dem Ziel-Anlagenbild als Kopie des Prototyps eingehängt. Nun muß diese Kopie mit den echten (Prozeß-) Variablen aus dem Variablenhaushalt verbunden werden.

Schritt	Typ	Projektierung
5	Variable	Definieren Sie zu der im Schritt 1 definierten Datenstruktur eine (Prozeß-) Variable, die für das Anwenderobjekt verwendet werden soll.
6	Instanz anlegen	<p>Kopieren Sie das Prototyp-Objekt aus der Grafikbibliothek mittels Drag&Drop in das Anlagenbild.</p> <p>Verbinden Sie dieses Objekt mit der (Prozeß-) Variablen über den Dynamic-Wizard Prototyp fest instanzieren:</p> <p>Der Wizard verbindet automatisch alle benötigten Strukturkomponenten der Variablen mit der richtigen Eigenschaft des prototypischen Bildbausteins, indem jede prototypische Variablenanbindung an jeder Eigenschaft durch die konkrete Variablenanbindung ausgetauscht wird.</p> <p>Nun ist ein Objekt entstanden, das zur Laufzeit mit den aktuellen Variablenwerten aktualisiert wird.</p>

3.3.11.4 Dynamische Instanz

Neben dem Dynamic-Wizard *Prototyp fest instanzieren* existiert auch noch ein Wizard für *Prototyp dynamisch instanzieren*. Was ist der Unterschied zu fest instanzieren und welche Schritte müssen modifiziert werden?

Im Gegensatz zum festen Verbinden eines Objektes mit Variablen können die Bildbausteine auch dynamisch verbunden werden. Dies bedeutet, daß abhängig von dem aktuellen Inhalt einer Variable die Instanz zur Laufzeit erst gesetzt wird. Zum Beispiel wird der obige Bildbaustein nicht fest mit der Variable verbunden, sondern es wird der Name der Variable dynamisch gehalten. Der aktuelle Name der Variable muß dann über eine Textvariable ermittelt werden. Diese Textvariable, die den jeweils aktuellen Namen der Variablen enthält, muß mit dem Bildbaustein verbunden werden.

Im Gegensatz zur festen Instanzierung müssen bei der Projektierung folgende Schritte geändert werden:

Schritt	Typ	Projektierung
2	Bildbaustein	Projektieren Sie über den <i>Graphics Designer</i> ein Anwenderobjekt mit den benutzerdefinierten Eigenschaften wie oben beschrieben. Das Anwenderobjekt muß eine Komponente <i>Statischen Text</i> beinhalten, dessen Eigenschaft <i>Text</i> als benutzerdefinierte Komponente übernommen wird. Diese Eigenschaft <i>Text</i> erhält als Attributnamen <i>Tagname</i> . Dieser Tagname wird für die dynamische Anbindung der (Prozeß-)Variablen verwendet.
5	Variable	Definieren Sie zu der im Schritt 1 definierten Datenstruktur eine (Prozeß-) Variable, die für das Anwenderobjekt verwendet werden soll.
6	Instanz anlegen	Kopieren Sie das Prototyp-Objekt aus der Grafikkbibliothek mittels Drag&Drop in das Anlagenbild. Verbinden Sie dieses Objekt mit der (Prozeß-) Variablen über den Dynamic-Wizard Prototyp fest instanzieren: Der Wizard verbindet automatisch alle benötigten Strukturkomponenten der Variablen mit der richtigen Eigenschaft des prototypischen Bildbausteins, indem jede prototypische Variablenanbindung an jeder Eigenschaft durch die konkrete Variablenanbindung ausgetauscht wird. Nun ist ein Objekt entstanden, das zur Laufzeit mit den aktuellen Variablenwerten aktualisiert wird.

Beim Einsatz der dynamisierten Anwenderobjekte und Prototypen ist von seiten des Projektors darauf zu achten, daß die nötigen *C-Aktionen* bereits an den Objekten hinterlegt wurden. Diese dürfen nicht gelöscht werden, da sonst die gesamte Baustein-Funktionalität verlorengeht.

3.3.11.5 Prototypbilder

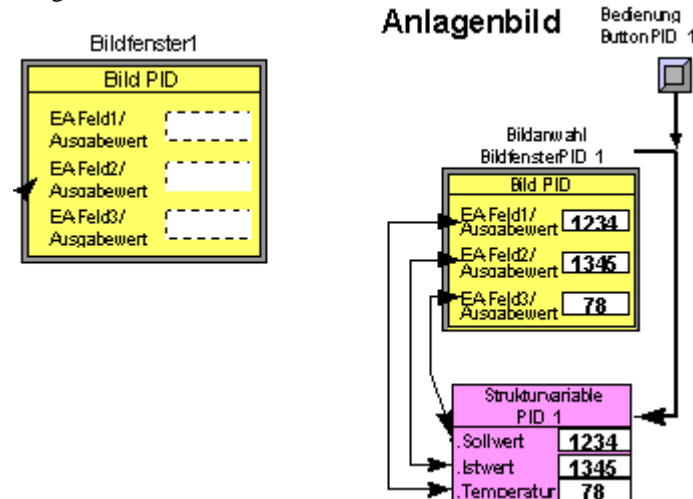
Die Technik der Prototypbilder geht noch einen Schritt weiter. Beim Einsatz von Prototypen kann das Konzept so flexibel aufgebaut werden, daß eine Änderung an dem Prototyp automatisch eine Nachführung an den erzeugten Objekten erfolgt. Diese Flexibilität erfordert sehr flexible C-Scripte, die angebunden werden müssen.

Die Technik der Prototypbilder arbeitet mit sogenannten Templatebildern, die mehrfach in ein oder mehrere Vaterbilder eingebunden werden. Ein Templatebild ist nur eine

Schablone, die erst in einem echten Objekt zum Leben erweckt wird. Ein Objekt auf Basis einer Schablone (=Prototypbild) entsteht durch eine sogenannte Instanzierung. Es können mehrere Instanzen (d.h. echte Objekte) zu einer Schablone erstellt werden.

Bildfenster Schablone

Eingesetztes Bildfenster mit aktuellen Daten aus dem Datenmanager anzeigen.



Nachträgliche Änderungen finden an zentraler Stelle (in der Schablone) statt und wirken sich auf alle Anwendungen (Instanzen) aus. Damit ist dies eine sehr effiziente Technik, die ein mühsames Nachziehen von Änderungen an vielen Stellen erübrigt.

In einem Vaterbild können bis zu 30 Instanzen (d.h. Objekte) jeweils eines Templatetyps eingebildet werden. Beim Einsatz unterschiedlicher Prototypen können auch mehr als 30 Objekte eingesetzt werden.

Die Prototypbilder sind Bildbausteine, die nach der Erstellung in die Bibliothek zur Wiederverwendung gelegt werden. Die eingesetzten Bildbausteine werden als Instanzen der Schablone in den Anlagenbildern verwendet. Diese Kopien zeigen die jeweils aktuellen Daten z.B. von Reglern oder Motoren, die in dem Baustein visualisiert werden. Die Darstellung der zugehörigen Regler- oder Motorkomponenten erfolgt automatisch.

Es können einfache, aber auch komplexe Bildbausteine gebildet werden. Ein Bildbaustein kann aus mehreren Komponenten bestehen. Diese sind teilweise oder ganz überlappt sind, aber eine zusammengehörige Einheit bilden. So können zum Beispiel alle Daten zu einem Motor, wie die Sicht auf den aktuellen Zustand, Verlaufsdaten, Wartungsdaten, etc. in einem Objekt zusammengefaßt und je nach Bedarf aktualisiert werden. Sind viele Motoren desselben Typs vorhanden, brauchen nach dem einmaligen Erstellen des Bildbausteins nur noch Kopien gebildet werden. Alles andere läuft automatisch.

Im einzelnen sind folgende Schritte für die Erstellung notwendig:

Schritt	Typ	Projektierung
1	Datenstrukturen	Definieren Sie die im Bildbaustein einzusetzenden Datenstrukturen über den Variablenhaushalt.
2	Bildfensteraufbau	Projektieren Sie über den <i>Graphics Designer</i> den Inhalt des Bildbausteines, z.B. Balken, EA-Felder etc.
3	Datenstruktur und Bild-	Verbinden Sie die Bildkomponenten mit den einzelnen Datenstruktur-Komponenten zusammen mit speziellen C-

Schritt	Typ	Projektierung
	fensteraufbau verbinden	Scripten (aus Beispiel-Projekt).
4	Bildfenster	Bildbaustein mit Bildfensterobjekt verbinden
5	Grafikbibliothek	Bildfenster in Grafikbibliothek legen


Nun kann der Bildbaustein in den Anlagenbildern über das Holen aus der Grafikbibliothek eingesetzt werden.

Zunächst betrachten wir die Erstellung der Schablone (Templatebild).

Schritt	leer	Projektierung
1		Anlegen einer Variablenstruktur (Datentyp Strukturtyp) im Datenmanager; hierbei wird festgelegt, aus wievielen Variablen die Struktur aufgebaut ist (Membervariablen), wie diese heißen und von welchem Datentyp diese jeweils sind (BIT, SHORT, etc.). z.B. PID mit Sollwert, Istwert und Temperatur als Strukturkomponenten.
2		Erstellen eines Bildes, das als Bildbaustein genutzt werden soll. Typischerweise ist dies kleiner als die Bildschirmgröße und kann entsprechend dimensioniert werden. Jedes Bild, das einmal entstanden ist, kann für die Bildung eines Templates verwendet werden. Das Bild wird mit den Grafikeditierfunktionen erstellt und es werden darin Grafikvariablen, wie EA-Felder, Balken, Zustandsanzeigen etc. Plaziert, aber nicht mit Variablen verbunden. Interne Beziehungen (<i>Direktverbindung</i>) zwischen Grafikobjekten, wie z.B. die änderungsgesteuerte Übergabe des Ausgabewertes eines EA-Feld an einen Balken, werden in diesem Bild projektiert.
3		Nun werden die Grafikfelder mit den Strukturkomponenten der zugehörigen Variablenstruktur verbunden. Diese Verbindungsprojektion verbindet auf Typebene (nur Schablone) und noch nicht mit konkreten Pozeßobjekten. Zu diesem Zweck gibt es ein vorbereitetes Beispiel-Projekt auf der WinCC CD (\Samples). In der Projektbibliothek (\Template) dieses Projektes gibt es ein Anwenderobjekt <i>TemplateInit</i> , das die Verschaltung vornimmt. Es liegt in der Grafikbibliothek und kann von dort in ein Bild, das typisiert werden soll, per Drag&Drop gezogen werden. <i>TemplateInit</i> verfügt bereits über eine fertige Scriptlogik. Sie arbeitet mit einer sogenannten <i>ConnectionTable</i> , die in der Projektierung als Tabelle ausgefüllt wird und genau die vorgenannten, unterstrichenen Einträge enthält. Über diesen Weg wird die Verbindung zwischen den Eigenschaften und den Strukturkomponenten festgelegt. Der Aufbau dieser Verbindungen kann innerhalb eines Templates oder auch von außerhalb gesetzt werden. In der speziellen Projekt-Grafikbibliothek liegen hierzu Anwenderobjekte, die rein äußerlich wie Bedientasten


Schritt	leer	Projektierung
		aussehen, die aber parametrierbare Information über die aufzurufende Schablone halten. Die gesamten Scripte für die Realisierung dieser bereits vorbereiteten Prototyperstellung müssen in das Ziel-Projekt geladen werden. Siehe dazu die abschließenden Schritte am Ende des Abschnittes unter Schritt 8-10. Ohne diese Projektfunktionen sind diese Prototypen nicht realisierbar.
4		Dieses Bild soll als Prozeßbox verwendet werden. Erstellen Sie dazu in einem temporären Bild (d.h. es wird nur für diesen Zwischenschritt benötigt) ein Bildfenster-Objekt und verbinden Sie die Eigenschaft <i>Bildname</i> dieses Objektes mit dem Bild, das den Bildbaustein enthält.
5		Legen Sie dieses Bildfenster-Objekt mittels Drag&Drop in die Grafikbibliothek.

Diese Schablone kann nun mehrmals in den Anlagenbildern eingesetzt werden. Die Verbindung zu den Prozeßvariablen erfolgt automatisch über die Namensgebung.

Schritt	Typ	Projektierung
6	Variable definieren	Definieren Sie eine (Prozeß-) Variable zu dem zugehörigen Datentyp (z.B. PID_1 vom Typ PID).
7	Instanzen anlegen	Kopieren Sie den Bildfenster-Baustein aus der Grafikbibliothek Vergeben Sie den Bildfenster-Objektnamen gleich dem Namen der (Prozeß-) Variablen, die Verwendung finden soll (z.B. PID_1): <i>Bildfensterobjekt</i> → <i>Bildfenster</i> → <i>Objektnamen</i> → <i>Statik</i>  auf PID_1 setzen

Beim Plazieren erhält der Bildbaustein den Namen einer strukturierten (Prozeß-) Variable, deren Werte die Zustandsdaten eines Prozeßobjektes enthalten. Zur Laufzeit versorgt sich der Bildbaustein dann automatisch mit den Zustandsdaten.

Für diese Prototypbilder werden einige C-Scripte verwendet, die als Projektfunktionen abgelegt werden bzw. wurden. Um die bereits vorbereiteten C-Scripte verwenden zu können, müssen folgende Scripte aus dem Beispielprojekt übernommen werden. Gehen Sie wie folgt vor:

Schritt	Typ	Projektierung
8	Kopieren der Funktionsdateien	Kopieren Sie aus dem Beispielsprojekt vom Pfad <Projektpfad>\Library alle benötigten Funktionen (.fct), z.B. LinkConnectionTable.fct in Ihr Projektverzeichnis <Projektpfad>\Library.
9	im Projekt bekanntgeben	Rufen Sie in WinCC den Global Script Editor (<i>Global Script</i> →  → <i>Öffnen</i>) auf, um diese neuen Funktion im folgenden bekanntzugeben: über die Bedientaste <i>Header generieren</i> können Sie nun die neuen Funktionen im Funktionsbaum der Projektfunktionen bekanntgeben. Die neuen Funktionen sind nun in der Liste der Projektfunktionen sichtbar.

Schritt	Typ	Projektierung
10	Anwender-objekte übernehmen	<p>Die Anwenderobjekte werden über die Projektbibliothek zur Verfügung gestellt. In einem neuen Projekt, in dem Sie bisher keine eigenen Symbole in der Bibliothek abgelegt haben, können Sie die Bibliothek aus dem Beispielprojekt einfach in Ihr Projekt kopieren:</p> <p><Projektpfad>\Library\library.pxl → in Ihren Pfad <Projektpfad>\Library kopieren!</p> <p>Ansonsten übertragen Sie die Anwenderobjekte über den Export-Mechanismus von einem Projekt in das andere Projekt: Exportieren Sie die gewünschten Symbole im Beispielprojekt als .emf-Dateien (<i>Datei</i> → <i>Export</i>) und importieren Sie diese .emf-Symbole im eigenen Projekt durch <i>Einfügen</i> → <i>Import</i> in ein temporäres Bild. Übertragen Sie die Symbole in Ihre Projekt-Bibliothek mittels Drag&Drop. Nutzen Sie dazu auch eine eigene Mappe z.B. Template.</p>

Bildbausteine (Typisierte Bildanteile oder Templates) bieten einen sehr großen Einsparungseffekt. Sie sind als Objekte mit Variablenreferenzen modelliert und werden z.B. in die Grafikbibliothek abgelegt. Sie werden aus dieser Grafikbibliothek entnommen, im Anlagenbild plaziert und zur Laufzeit automatisch mit Daten versorgt. Eine Verbindungsprojektierung von einzelnen Variablen mit Grafikanteilen, wie E/A-Felder, Balken, etc. ist nicht mehr notwendig.

Beim Einsetzen dieser Prototypbilder gibt es unterschiedliche Wege, wodurch die Bildbaustein-Komponenten mit den aktuellen Namen versorgt werden. Dies erfolgt über folgende Varianten, die im folgenden kurz genannt werden sollen:


- der Instanzname wird bei **Anwahl** des Bildfensters selbst ermittelt: dazu wird eine vorgegebene Projektfunktion (EnabelTemplateInstance) bei dem Ereignis Bildanwahl im Bildfenster selbst hinterlegt.
- Es wird der Instanzname über eine **Ein-/Ausgabevariable** festgelegt, die automatisch über ein Script im Bildfenster gelesen und der Instanzname somit ermittelt wird, hierzu wird das vorbereitete Anwenderobjekt *InstanceCallButton+Template* eingesetzt.
- Eine **Bedientaste** übergibt den Instanznamen direkt an das aufgerufene Bildfenster: hierzu wird das vorbereitete Anwenderobjekt *InstanceCallButtons+Template* eingesetzt.

Zu diesen Varianten finden Sie in dem Beispielsprojekt ausführliche Beispiele auf der WinCC CD (\Samples).

3.3.11.6 OCX-Objekte

OCX- oder ActiveX - Objekte sind Bildbausteine, die als ablauffähige Komponenten zur Verfügung stehen. Standardmäßig werden einige von WinCC angeboten, z.B. die WinCC Digital/Analog Clock Control.

Diese Bausteine können sehr einfach in die Anlagenbilder eingebracht werden.

Schritt	Typ	Projektierung
1	OCX-Objekt einhängen	Wählen Sie in der Objektpalette im <i>Graphics Designer</i> unter den Smart-Objekten den Typ <i>OLE-Control (OCX)</i> aus. Ziehen Sie im Anlagenbild das Objekt mittels gedrückter  auf und wählen Sie in der anschließenden Dialogbox das gewünschte Element aus.
2	Eigenschaften verbinden	Das eingehängte Objekt hat ebenfalls Eigenschaften und Ereignisse. Welche Eigenschaften bzw. Ereignisse zur Verfügung stehen sind von dem spezifischen OCX selbst abhängig. Eine Verbindung mit einer Variablen kann z.B. mit der Eigenschaft Prozeßanschluß vorgenommen. werden.

Erstellung

Die OCX-Bildbausteine müssen mit einer eigenen Entwicklungsumgebung erstellt werden. Dies ist zum Beispiel Microsoft Visual C++ 5 oder Microsoft Visual Basic 5. Über diesen Weg werden die Bildbausteine geändert bzw. verbessert. Die OCX-Bausteine sind sehr performant, können aber mit WinCC Projektierungsmitteln nicht erstellt werden. In diesem Fall ist man immer auf eine externe Erstellung bzw. Änderung angewiesen. Die Technik der Prototypbilder kommt im Gegensatz zu der änderungsfreundlichen und performanten OCX-Programmierung mit reinen WinCC-Mitteln aus. Das heißt, das Know how um die OCX-Programmierung ist dabei nicht notwendig.

Es werden heute bereits eine Vielzahl an derartigen Bausteinen angeboten. Unter anderem werden in Zusammenhang mit der Integration der beiden Welten Bedienen&Beobachten als auch der SPS-Programmierung (PCS7) im Anlagenbereich fertige Bausteine als PCS7-Faceplates angeboten.

Registrierung

Die erstellten bzw. zugekauften OCX-Bausteine müssen auf der jeweiligen WinCC-Station registriert werden. Welche OCX-Objekte auf der WinCC-Station verfügbar sind, kann bereits im Auswahl-Dialog im *Graphics Designer* (siehe obige Beschreibung) ersehen werden. Alle auf der Maschine registrierten OCX-Elemente werden in dem Dialog aufgeführt. Ein OCX-Element ist in Form einer Datei mit der Extension .OCX oder .dll auf der Maschine abgelegt.

Wurde ein Baustein noch nicht registriert, so kann dies in dem OLE-Control-Dialog von WinCC ebenfalls erfolgen. Die Dialog-Box enthält eine Bedientaste für das Registrieren als auch für das Entfernen der Registrierung der aktuell angewählten Komponente. Für den Registriervorgang muß die zugehörige Datei auf der WinCC-Station vorliegen.

Die Verträglichkeit bzw. die Funktionsweise der OCX-Komponenten muß von dem Projektteur selbst getestet werden. Nur die mit WinCC gekennzeichneten OCX-Bausteine sind in der WinCC-Umgebung eingesetzt und getestet worden.

3.3.12 Online Projektierung (Runtime) - Hinweise, Einschränkungen

Bei der online Projektierung gibt es einige Dinge zu beachten.
Aus verschiedenen Gründen können einige wenige Änderungen online nicht oder nur unter bestimmten Voraussetzungen durchgeführt werden oder die Änderungen werden erst zu einem späteren Zeitpunkt wirksam.

Control Center

Folgende Änderungen werden nicht übernommen:

- Änderung des Typs eines Rechners in der Rechnerliste

Im Runtime sind folgende Projektierungsschritte nicht möglich:

- Löschen/Umbenennen von Variablen
- Änderung des Datentyps einer Variablen

Alarm Logging

Folgende Änderungen werden nicht übernommen:

- Änderung der Archive / Protokolle
- Änderung der Sammelmeldungen
- Jede Meldung die über eine gesamte Summe von 500 Einzelmeldungen bei laufendem Runtime

Im Runtime sind folgende Projektierungsschritte nicht möglich:

- keine Einschränkungen

Tag Logging

Folgende Änderungen werden nicht übernommen:

- keine Einschränkungen.

Im Runtime sind folgende Projektierungsschritte nicht möglich:

- Tabellen der Anwenderarchive können angelegt, aber nicht verändert werden.
- Das Löschen von Daten in Tag Logging und Anwenderarchiven.

Ausnahmen bei der Projektierung im Runtime:

- Über das Runtime-API von Tag Logging können die Tabellen der Anwenderarchive bearbeitet und gelöscht werden.

Global Script

Folgende Änderungen werden nicht übernommen:

- Die Änderung eines Wizard Scriptes werden erst nach Neustart vom *Graphics Designer* übernommen.
- Geänderte Wizard Scripte.

Im Runtime sind folgende Projektierungsschritte nicht möglich:

- keine Einschränkungen

Report Designer

Folgende Änderungen werden nicht übernommen:

- Änderungen am Meldefolgeprotokoll, da es im Runtime einmal gestartet immer aktiv bleibt und die Layoutinformationen nicht wieder neu lädt.

Im Runtime sind folgende Projektierungsschritte nicht möglich:

- keine Einschränkungen.

Redundancey

Folgende Änderungen werden nicht übernommen:

- Der Rechnername des Partners kann nicht auf einen dritten Rechner umgeschaltet werden.
- Der AutoSwitcher kann nicht geändert werden, d.h. man muß am Anfang projektieren, wohin der AutoSwitcher umschalten soll. Er schaltet allerdings auch zurück falls der andere ausfällt.

Im Runtime sind folgende Projektierungsschritte nicht möglich:

- keine Einschränkungen

SIMATIC S7 Protokoll Suite oder S/7PMC Kanal

Folgende Änderungen werden nicht übernommen:

- Alle Diagnoseparameter über die S7Chn.ini (nicht veröffentlicht) werden nicht online übernommen.
- Alle Änderungen an Kommunikationsadressen werden zwar online übernommen, werden aber nur bei einem Verbindungsaufbau ausgewertet.

Im Runtime sind folgende Projektierungsschritte nicht möglich:

- keine Einschränkungen

Text Library

Folgende Änderungen werden nicht übernommen:

- keine Einschränkungen.
 - In der Text Library werden die geänderten Texte mit *Datei → Änderungen an aktives Projekt senden* übernommen
 - Im Alarm Logging werden die Änderungen in die Textbibliothek mit *Datei → Speichern* übernommen.

Im Runtime sind folgende Projektierungsschritte nicht möglich:

- keine Einschränkungen

User Administrator

Folgende Änderungen werden nicht übernommen:

- Änderungen der Benutzerberechtigung werden erst nach erneutem Login bzw. Logoff wirksam.

Im Runtime sind folgende Projektierungsschritte nicht möglich:

- keine Einschränkungen

4 WinCC - C-Kurs

Aus den bisherigen Erfahrungen stellt sich die Script Sprache von WinCC den C Neulingen erst einmal als eine nur schwer zu lösende Aufgabe dar. Für viele Anwendungen ist es ausreichend, wenn Sie die vorhandenen Funktionen mit den entsprechenden Parametern versorgen und verwenden.

Hinweis:

Für das Dynamisieren von Objekten ist es nicht unbedingt notwendig, eigene Funktionen oder Aktionen zu erstellen, da für die Dynamisierung ein entsprechender *Dynamik-Dialog* zur Verfügung steht. Dieser Dialog erzeugt ablauffähige und erweiterbare *C-Aktionen*. *Dynamik-Dialoge* laufen in einem speziellen Speicherbereich des Betriebssystems ab und sind dadurch deutlich schneller als *C-Aktionen*. Der Funktionsumfang der *Dynamik-Dialoge* ist jedoch begrenzt. Ein *Dynamik-Dialog* kann nachträglich, allerdings mit Performanceverlust, in eine *C-Aktion* umgewandelt werden.

Die schnellste Art der Dynamisierung ist jedoch die *Direktverbindung*. Aber auch hier gelten gegenüber den *C-Aktionen* bestimmte Einschränkungen (z.B. nur für Objekte im Bild).

Anwendern mit sehr anspruchsvollen Aufgaben steht mit den *C-Aktionen* im *Graphics Designer* und im *Global Script Editor* eine sehr leistungsfähige Scriptsprache zur Verfügung. Sollten in Ihrer ersten Euphorie beim Entwickeln die Scripte einmal etwas größer und komplexer ausfallen, so können Sie diese als ausführbare *dll (Dynamic Link Library)* Funktionen, mit deutlichem Performancegewinn, in WinCC einbinden.


Zielgruppe

C-Neulingen soll dieser C-Kurs die wichtigsten Themen der Sprache C veranschaulichen. All denen, die bereits in C programmieren, soll dieser Kurs die Besonderheiten der Entwicklungsumgebung näher bringen.


Die Beispiele für diesen Kurs sind im WinCC Projekt *Cours_00* projektiert.

Startbild vom C-Kurs



Für jedes Thema in diesem Kurs haben wir ein Bild mit mehreren Beispielen projiziert. Die einzelnen Bilder erreichen Sie über die Auswahl der entsprechenden Button mit der  in der Tastenleiste.

Mit dem Button *Diagnose* schalten Sie das WinCC Diagnosefenster ein / aus. In den *C-Aktionen* werden alle Ausgaben mit *printf* in dieses projizierte Diagnosefenster ausgegeben.

Wird bei den einzelnen Beispielen der entsprechende Button mit der  **R** angewählt, so wird der Sourcecode zum entsprechenden Beispiel angezeigt.

4.1 Die Entwicklungsumgebung für Scripte in WinCC

WinCC stellt Ihnen mit den *C-Aktionen* im *Graphics Designer* und dem *Global Script Editor* eine sehr leistungsfähige Scriptsprache zur Verfügung. Die Syntax der Scriptsprache entspricht Standard C (genormt nach ANSI) und wird interpretativ abgearbeitet. Dies bedeutet, daß die in der Scriptsprache formulierten Anweisungen beim Übersetzen nur syntaktisch geprüft, aber nicht sofort ausgeführt werden.

Global Script ist der Oberbegriff für *C-Funktionen* (*Standard- und Projektfunktionen*) und *globalen Aktionen*, die je nach Typ, projektweit oder auch projektübergreifend verwendet werden können.

4.1.1 Funktionen und Aktionen in WinCC

C-Aktionen (Funktionen und Aktionen) werden zur Laufzeit (Runtime) ausgeführt und dienen der Prozeßvisualisierung und der Prozeßführung. Mit Funktionen und Aktionen können Objekteigenschaften verändert werden und man kann auf Ereignisse reagieren.

Der Aufbau einer Funktion:

Funktionen bestehen aus zwei unterschiedlichen Bereichen:

Kopf der Funktion	Typ des Rückgabewertes Funktionsname (Typ des Parameters1 Parameternamen1,)
Rumpf der Funktion	{ Variablen deklarieren; Befehle; return (Rückgabewert); } // Funktionsende

Der Funktionskopf

Er beschreibt die Umgebung der Funktion (Schnittstelle zur Außenwelt)

Ein Beispiel für einen *Funktionskopf*:

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
char* lpszPropertyName, UINT nFlags, int x, int y)
{
```

Im Kopf der *Funktion* werden Name (wie wird die Funktion aufgerufen?) und Übergabeparameter (welche Daten werden an die *Funktion* übergeben und welcher *Rückgabewert* liefert die Funktion) der Funktion beschrieben.

Der *Funktionskopf* wird beim Aufruf der *C-Aktion* automatisch angezeigt und muß vom Projektteur nicht geschrieben werden.

Der *Funktionskopf* zeigt dem Projektteur, welcher *Rückgabewert* zurückgeliefert werden muß und welche Daten innerhalb der Funktion bereits zur Verfügung stehen.

Abhängig von der Dynamisierung einer Eigenschaft bzw. eines Ereignisses werden unterschiedliche Funktionsköpfe eingeblendet.

Funktionskopf am Ereignis

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
//Funktionsrumpf
}
```

Der *Rückgabewert* ist bei diesen Aktionen immer vom Typ *void* (d.h. nichts). Der Befehl *return;* kann in den *C-Aktionen* weggelassen werden, da dieser automatisch am Ende einer *C-Aktion* ausgeführt wird.

Funktionskopf an einer Eigenschaft

```
#include "apdefap.h"
long _main(char* lpszPictureName, char* lpszObjectName, char*
           lpszPropertyName)
{
//Funktionsrumpf mit Rückgabewert
}
```

Bei den *C-Aktionen* für Objekt-Eigenschaften wird die neue Einstellung der zu dynamisierenden Eigenschaft über den aktuellen *Rückgabewert* der C-Aktion geliefert. Der Typ des *Rückgabewertes* z.B. eine Zahl (in unserem Beispiel *long*) oder eine Zeichenfolge (z.B. *char **) wird im Kopf der *C-Aktion* definiert:

Der Funktionsrumpf

Er beschreibt den Inhalt der Funktion (**was tut die Funktion?**)

```
/* Deklaration der Variablen */
int number1;

/* Wertzuweisungen zu den Variablen */
number1=123;

/* Ausgabe der Variablenwerte im Diagnosefenster */
printf("Die erste Zahl ist %d\n",number1);
}
```

Im *Funktionsrumpf* werden die einzelnen Befehle aufgeführt. Der *Funktionsrumpf* wird durch die öffnende geschweifte Klammer eingeleitet. Die schließende, geschweifte Klammer beendet diesen wieder.

Im *Funktionsrumpf* werden zusätzliche Daten, die während der Bearbeitung der *Funktion* benötigt werden, definiert (lokale Daten). Diese Daten werden am Ende der *Funktion* automatisch gelöscht.

Die einzelnen Befehle im *Funktionsrumpf* werden jeweils mit Semikolon abgeschlossen.

Rückgabewert

Der *Rückgabewert* der *Funktion* (z.B. die neue Farbe oder neue PositionX) wird am Ende der Funktion im *Funktionsrumpf* zurückgeliefert.

Für die Formulierung der C-Sequenzen wird auf bereits existierende Funktionen zurückgegriffen. Diese Funktionen können über den Funktionsbaum eingblendet und genutzt werden. Diese projektweiten und projektübergreifenden Funktionen werden im *Global Script Editor* formuliert. Dabei werden folgende Typen unterschieden:

Standard-Funktionen

Sie können neue *Standard-Funktionen* erzeugen und vorhandene ändern.

Die *Standard-Funktionen* sind projektübergreifend bekannt und in Kategorien eingeteilt.

Die Bearbeitung erfolgt im *Control Center* mit dem Editor *Global Script*.

Standard-Funktionen werden zur Dynamisierung von *grafischen Objekten* und *Archiven* verwendet. Der Aufruf von *Standard-Funktionen* kann in *Projekt-Funktionen*, in anderen *Standard-Funktionen* und in *Aktionen* erfolgen.

Projekt-Funktionen

Sie können neue *Projekt-Funktionen* erzeugen und vorhandene ändern.

Die *Projekt-Funktionen* sind nur in dem *Projekt* bekannt, in dem sie erzeugt wurden. Das ist **der einzige Unterschied** zu den *Standard-Funktionen*.

Die Bearbeitung erfolgt im *Control Center* mit dem Editor *Global Script*.

Aktionen, die mehr als einmal verwendet werden, sollten als *Projekt-Funktionen* oder *Standard-Funktionen* definiert werden.

Projekt-Funktionen können und sollten analog der *Standard-* und *Internen Funktionen* kategorisiert werden.

Vorteile von *Projekt-Funktionen* sind:

- ihre **Wiederverwendbarkeit**. Wird die *C-Aktion* einmal projektiert und umfangreich getestet, kann diese jederzeit ohne zusätzlich Projektierung und ohne erneute umfangreiche Tests genutzt werden.
- einfache **Änderung**. Wird eine Änderung an der *C-Aktion* vorgenommen, so erfolgt die Änderung nur an einer einzigen Stelle – in der *Projekt-Funktion*. Die projektierten Dynamisierungen in allen Bildern müssen nicht mehr gesucht und angepaßt werden. Dies vereinfacht nicht nur die Projektierung, sondern auch die Wartung und die Fehlersuche.
- ein **reduziertes Bildvolumen** und dadurch schnellere Bildaufschlagzeit. Werden in den Bild-Aktionen Aufrufe von Projekt- oder Standard-Funktionen eingesetzt, so wird das gesamte Bildvolumen reduziert.
- Projektfunktionen können durch Paßwort (**Benutzerberechtigung**) vor Änderungen geschützt werden (Schutz von Projektierungsdaten und Projektierungs Know-how)

Interne-Funktionen

Interne Funktionen können nicht erstellt oder geändert werden und sind projektübergreifend bekannt.

Interne-Funktionen verwenden Sie zur Dynamisierung von *grafischen Objekten, Archiven, in Projekt-Funktionen, in Standard-Funktionen* und in *Aktionen*.

Interne Funktionen sind in Kategorien unterteilt.

Aktionen

Aktionen (globale und am Objekt) können erstellt und geändert werden. Diese Aktionen sind nur in dem Projekt bekannt, in dem sie erstellt worden sind. Projekt-, Standard-, und interne Funktionen verwenden Sie:

- in *Aktionen* am Objekt.
- in *Aktionen*, die im *Dynamik-Dialog* erstellt werden.
- zur Dynamisierung von *Alarmen*, *Prozeßwert-*, *Verdichtungs-* und *Anwenderarchiven*.

Hinweis:

Aktionen werden interpretativ abgearbeitet. Bei Einsatz zahlreicher oder umfangreicher Aktionen ist deshalb mit einer höheren Systembelastung zu rechnen. Umfangreiche Aktionen sollten daher besser durch eigene *DLLs* (*D*ynamic *L*ink *L*ibraries) ersetzt werden.

Aktionen verwenden Sie zur Laufzeit (Runtime) in der Prozeßführung. Sie werden durch einen *Trigger* zur Ausführung gebracht. Hierzu ist das Projekt zu aktivieren.

Eine globale Aktion erstellen Sie mit dem Editor *Global Script*. Starten Sie diesen Editor im *Control Center*.

4.1.2 Die Editoren für Funktionen und Aktionen

Funktionen und globale Aktionen werden mit dem Editor *Global Script* bearbeitet. Funktionen unterscheiden sich in Projekt- und Standard-Funktionen. Diese Funktionen werden in Bildern objektbezogen als *C-Aktion* am Objekt oder im Meldesystem verwendet. Die globalen Aktionen dagegen sind objektunabhängig und werden nur bei bestimmten Ereignissen (Trigger über Zeit oder Variable) bearbeitet.

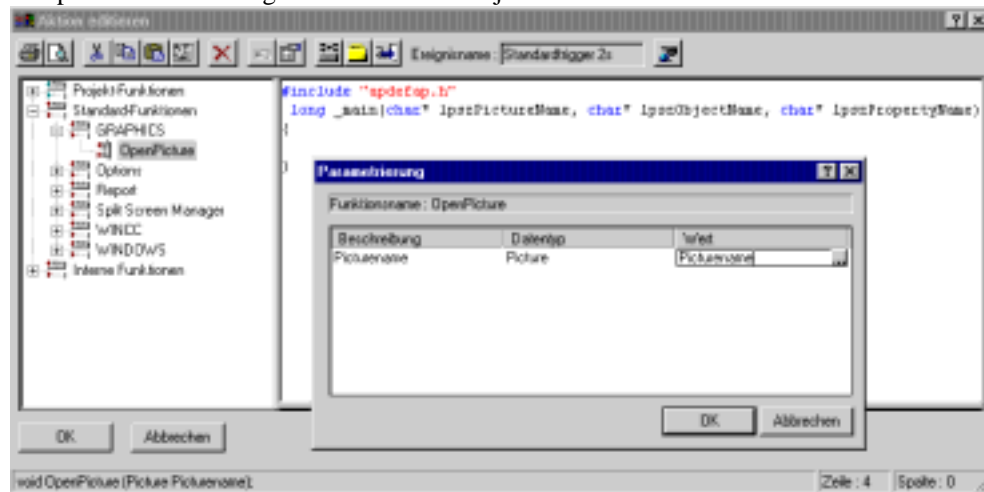
Die Unterschiede dieser beiden Editoren bestehen in der Art des Aufrufs und in ihrem Funktionsumfang.

Der Editor für Aktionen am Objekt

Der Editor wird im *Graphics Designer* bei der Dynamisierung der *Objekte* verwendet. Mit diesem Editor kann nur der *Funktionsrumpf* bearbeitet werden. Der *Funktionskopf* wird beim Aufruf automatisch erzeugt.

Im folgenden Bild wird die Parametrierung der eingefügten Funktion über den entsprechenden Dialog durchgeführt.

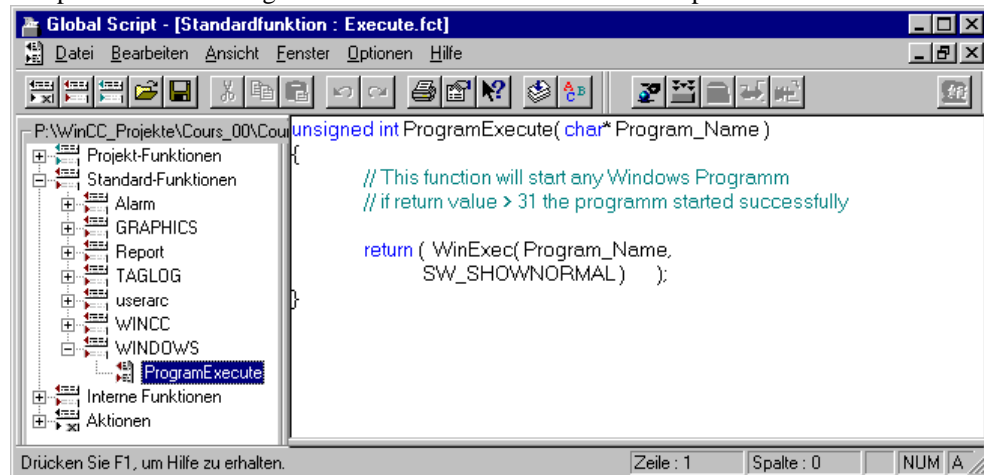
Beispiel der Bearbeitung einer Aktion am Objekt



Der Editor Global Script

Der Editor wird im *Control Center* aufgerufen. Es können *Funktionen* und *Aktionen* bearbeitet werden. *Aktionen* am *Objekt* können mit diesem Editor aber nicht bearbeitet werden. Bei der Bearbeitung von *Aktionen* ist der *Funktionskopf* geschützt, das heißt, er kann nicht verändert werden. Bei *Funktionen* kann der *Funktionskopf* und der *Funktionsrumpf* bearbeitet werden.

Beispiel der Bearbeitung einer Standardfunktion im Global Script.



Die nachfolgend aufgelisteten Bedienfunktionen gelten für beide Editoren.

Die Tastenbedienung im Editor

Funktion	Tastenbedienung
neue Zeile	RETURN
Zeichen löschen rechts	DEL
Zeichen löschen links	Backspace
Sprung an den Anfang der Zeile	HOME
Sprung an das Ende der Zeile	END
Sprung an den Textanfang	Strg+HOME
Sprung an das Textende	Strg+END
Bewegen der Einfügemarke	mit den Cursortasten
Markierten Text ausschneiden	Strg+X
Markierten Text kopieren	Strg+C
Text aus der Zwischenablage einfügen	Strg+V

Die Mausbedienung im Editor

Funktion	Mausbedienung
Text markieren	mit linker Maustaste
Ein Wort markieren	Doppelklick mit linker Maustaste
Einfügemarke versetzen	mit linker Maustaste

Weitere Editierfunktionen sind:

Funktion	Erklärung
Schreibmodus	Ist Einfügen
Markierter Text	Dieser wird durch das nächste mit der Tastatur eingegebene Zeichen ersetzt
Einen Bereich markieren	Einfügemarke an den gewünschten Beginn der Markierung setzen, Umschalt-Taste gedrückt halten, Einfügemarke an das gewünschte Ende der Markierung setzen
Eine Markierung erweitern	Umschalt-Taste gedrückt halten, Einfügemarke an das gewünschte Ende der Markierung setzen

4.1.3 Funktionen und Aktionen erstellen

Eine neue Funktion erstellen

Folgende Schritte sind für die Erstellung einer Standard- oder Projekt-Funktion notwendig:

Schritt	Vorgehen: Standard- oder Projektfunktion erstellen
1	Funktion formulieren
2	Funktions-Informationen ergänzen
3	Funktion übersetzen
4	Funktion speichern und gegebenenfalls umbenennen
5	gegebenenfalls Header-Dateien generieren

Damit haben Sie Ihre Standard- oder Projekt-Funktion erstellt.

Die Header-Datei *apdefap.h* bindet die Header-Datei *ap_glob.h* ein, welche die Deklaration der *Standard-Funktionen* beinhaltet.

Die Dateien *apdefap.h* und *ap_glob.h* befinden sich im Verzeichnis *<WinCC-Installationsverzeichnis>\APLIB*.

Wird eine neue *Standard-Funktion* angelegt und werden dort andere *Standard-Funktionen* genutzt, so muß die Header-Datei *apdefap.h* oder *ap_glob.h* eingebunden werden.

Neue *Standard-Funktionen* werden zu den vorhandenen *Standard-Funktionen* auf dem *Serverrechner* im Verzeichnis *<WinCC-Installationsverzeichnis> \APLIB* abgelegt.

Die im Projekt definierten *Projekt-Funktionen* dagegen werden auf dem *Serverrechner* im Projektpfad unter *<Projektname> \Library* abgelegt. Die Funktionsköpfe (Funktionsprototypen genannt) der neuen *Projekt-Funktionen* werden in der Deklarationsdatei (für C-Funktionen) *Ap_pbib.h* im Verzeichnis *<Projektname> \Library* hinterlegt.

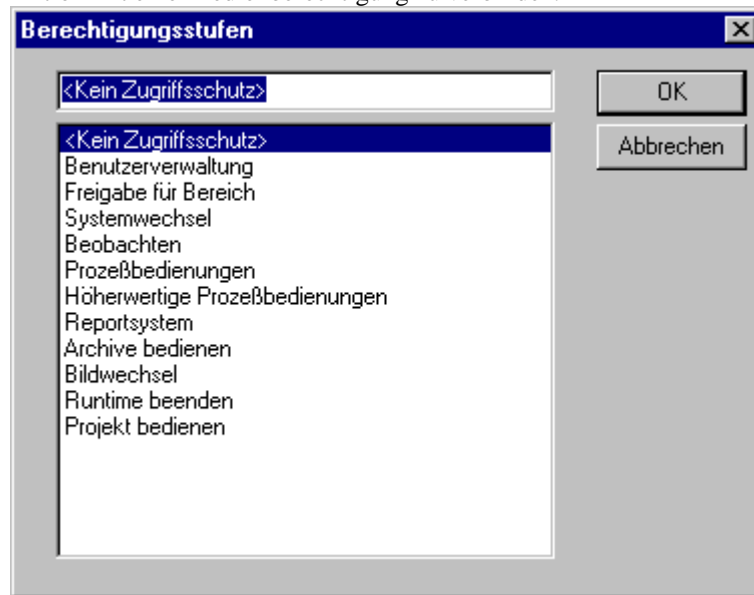
Eine neue Aktion erstellen

Folgende Schritte sind zur Erstellung einer globalen Aktion notwendig:

Schritt	Vorgehen: Aktion erstellen
1	Aktion formulieren
2	Aktions-Informationen ergänzen
3	den oder die Trigger als auslösendes Ereignis festlegen
4	Aktion übersetzen
5	die Bedienberechtigung einstellen
6	Aktion speichern und gegebenenfalls umbenennen

Damit haben Sie eine globale Aktion erstellt.

In *Global Script* haben Sie die Möglichkeit, mit *Bearbeiten* → *Bedienberechtigung* die Aktion mit einer Bedienberechtigung zu verbinden.



Beim Erstellen einer neuen Aktion wird automatisch die Header-Datei *apdefap.h* vom Projektverzeichnis ...*<Projektname>*\LIBRARY eingebunden.

Ist keine *Projekt-Funktion* vorhanden, so wird die Header-Datei aus dem Verzeichnis ...*APLIB* eingebunden.

Somit stehen in den Aktionen die *Standard-* und *Projekt-Funktion* zur Verfügung.

Bei Mehrplatzsystemen unterscheidet man zwischen globalen und lokalen Aktionen.

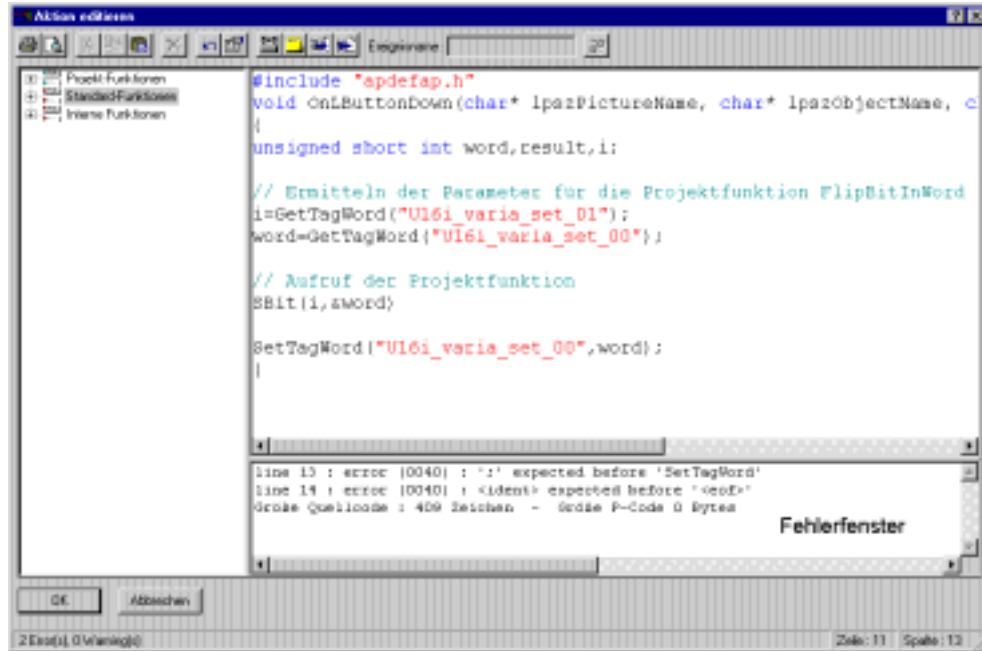
Globale Aktionen sind im Projektverzeichnis rechnerunabhängig abgelegt (...*<Projektname>*\PAS). Diese sind rechnerübergreifend aktiv.

Lokale Aktionen sind im Projektverzeichnis rechnerzugehörig abgelegt (...*<Projektname>**<Rechnername>*\PAS). Somit sind lokale Aktionen nur auf dem zugehörigem Rechner aktiv.

Wird das Runtime-Modul *Global Scripts* in die Anlaufliste aufgenommen, werden **alle** globalen Aktionen die zum Projekt gehören aktiviert. Das Aktivieren erfolgt beim Hochlauf in den Runtime-Modus.

4.1.4 Funktionen und Aktionen testen

Fehlerhinweise




Die in dem *Fehlerfenster* aufgeführten Syntax-Meldungen (*Errors* oder *Warnings*) können in jedem C-Buch bzw. Handbuch zu C-Compiler (z.B. Microsoft Visual C) nachgelesen werden. Es handelt sich um Standard-Fehlermeldungen, die jeder C-Compiler bei der Syntaxprüfung liefert.

Errors führen immer dazu, daß die Aktion nicht ablauffähig ist, während *Warnings* nur Hinweise sind, die auf eventuelle Fehler bei der Ausführung hinweisen.

Hinweis:

Die im Fehlerfenster gemeldeten Warnungen sollten auf jeden Fall beachtet werden. Eine Korrektur ist notwendig, um einen fehlerfreien Ablauf zu gewährleisten. Warnungen können zum Beispiel falsche Zahlenformate bei den Funktionsaufrufen sein. Außerdem ist die Beseitigung von Warnungen guter Programmierstil!

Beseitigen von syntaktischen Fehlern

Die jeweilige Fehlerzeile im *Fehlerfenster* des C-Aktionseditors beschreibt den Syntaxfehler im Detail mit Angabe der Codezeile. Mit  auf die Fehlerzeile wird die Schreibmarke automatisch an die entsprechende Stelle im Code gesetzt. Auf diesem Wege kann der Fehler an Ort und Stelle analysiert und ausgebessert werden.

Testausgaben der C-Aktion

Die Ausführung einer *C-Aktion* kann mittels eigener Ausgabebefehle getestet werden. Diese Ausgabebefehle sind mittels der Ausgabefunktion von C (*printf*) zu schreiben. Diese Ausgabeanweisungen werden bei Ablauf der Aktion im Diagnosefenster ausgegeben. Das Diagnosefenster wird im *Graphics Designer* über *Smart-Objekte* → *Applikationsfenster* → *Global-Script* → *GSC_Diagnose* im Bild projiziert.

4.1.5 Funktionen und Aktionen importieren, exportieren

Globale Aktionen können exportiert und importiert werden.
Eine importierte Aktion ersetzt die Aktion im aktiven Fenster vollständig.

C-Aktionen, die bereits formuliert wurden, können auf verschiedene Art und Weise wieder genutzt werden:

Ablegen der C-Aktion in der Zwischenablage

Wird die formulierte *C-Aktion* sofort wieder in der nächsten Aktionsmaske (einer weiteren Eigenschaft oder eines Ereignisses) genutzt, so kann die aktuelle *C-Aktion* bzw. Ausschnitte dieser *C-Aktion* über die Zwischenablage ex- und importiert werden. Hierzu stehen Ihnen die *Kopier/Einfüge-Taste* in der Symbolleiste oder die Tasten-Kurzbedienungen (*CTRL C*, *CTRL V*) zur Verfügung.

Export von C-Aktionen

Eine geschriebene *C-Aktion* kann jederzeit - auch für Kopiervorgänge (Kopieren von Aktionen an andere Eigenschaften oder Ereignisse) - in einer Datei, z.B. *Bildname.act* abgelegt werden. Hierzu wählen Sie die Taste *Aktion exportieren* und vergeben einen Dateinamen. Auf diese Weise können Basisaktionen, die immer wieder genutzt werden, jederzeit in dem Bildpfad des Projektes (*\\<Projektname>\GraCS*) abgelegt bzw. zur Verfügung gestellt werden.

Import von C-Aktionen

Eine vorhandene C-Aktion, die in dem Bildpfad des Projektes z.B. (*\\<Projektname>\GraCS*) zur Verfügung gestellt wird, kann jederzeit in der Aktionsmaske eingefügt und modifiziert werden. Wählen Sie die *Eigenschaft* welche Sie dynamisieren wollen und rufen den Editor für *C-Aktionen* auf. Wählen Sie die Taste *Aktion importieren* und selektieren Sie den Dateinamen, der die gewünschte Aktion enthält, z.B. *Bildname.act*. Auf diese Weise können Basisaktionen jederzeit genutzt und nachträglich an die Dynamisierung angepaßt werden.

Ablage eines projektierten und verbundenen Objektes als Bibliotheksobjekt

Objekte, die bereits mit (komplexeren) *C-Aktionen* vorprojektiert wurden, können als gesamtes Grafik-Objekt mit Dynamik in der *Projektbibliothek* zur weiteren Nutzung hinterlegt werden.

Hierzu muß das Objekt nur über *Drag&Drop* (Anwahl und Kopieren mittels der Mausbedienung) in die angewählte *Projektbibliothek* kopiert werden. Es werden außerdem die projektierten Aktionen zu den *Eigenschaften* bzw. *Ereignissen* übernommen. Ein Beispiel für bereits hinterlegte Objekte mit *C-Aktionen* finden Sie in der Mappe Anwenderobjekte.

4.2 Variablen in C




In unserem Projekt erreichen Sie die Beispiele zum Thema Variablen durch die Anwahl des Button *Variablen*. Mit der  wird der Sourcecode zum jeweiligen Beispiel angezeigt.

Bild für Variablen



Nach der Anwahl des Button *Variable* mit der  sehen Sie das oben dargestellte Bild. Die einzelnen Button im Bild führen die beschriebenen Beispiele aus. Mit der  wird der Sourcecode zum Beispiel angezeigt.

Bevor wir mit den Beispielen beginnen, etwas Theorie zum Thema Variablen.

Variablen bestehen aus einem *Variablenamen*, einem *Variablentyp* und dem Inhalt. Variablen kann man mit einem *Behälter* vergleichen. Mit den Variablenamen geben wir dem *Behälter* einen eindeutigen Namen, denn wir wollen ja den *Behälter* mit Inhalt später wieder lokalisieren und verwenden.

Diese C Variablen sollten Sie nicht mit WinCC Variablen verwechseln. Die hier beschriebenen Variablen stehen nur innerhalb der *C-Scripte* zur Verfügung.

Variablentypen in C

C kennt folgende Grundtypen von Variablen, die folgenden Wertebereichen entsprechen:

Typ	Wertebereich
int	Ganzzahlen (Zahlen ohne Nachkommaanteil) z.B. 2 / -3 / 5 / u.s.w.
char	ein Zeichen z.B. a / k / z , aber auch die RETURN-Taste
float	Fließkommazahlen (Zahlen mit Nachkommaanteil) z.B. 3.23 / 4.32 / 3.01
double	Fließkommazahlen mit doppelt so großem Wertebereich wie der Float-Typ

Der Typ *double* unterscheidet sich nur durch den Wertebereich vom Typ *float* . Dadurch können im Typ *double* die Zahlen genauer dargestellt werden (höhere Auflösung).

Dem Variablentypen *int* kann noch eines der Schlüsselwörter *signed* oder *unsigned* vorangehen. Steht *signed* davor, kann das Programm Vorzeichen unterscheiden, steht *unsigned* davor, dürfen nur positive Werte benutzt werden.

Beim Typ *int* kann zusätzlich noch *long* oder *short* vorangesetzt werden.

Wenn nicht explizit etwas anderes angegeben ist, sind alle *int* Variablen vom Typ *signed int*.

Hier nun die Wertebereiche der Variablentypen:

Typ	Wertebereich
int	- 2 147 483 648 bis 2 147 483 647
unsigned int in WinCC auch (DWORD)	0 bis 4 294 967 295
short int in WinCC auch (SHORT)	- 32 768 bis 32 767
long int	- 2 147 483 648 bis 2 147 483 647
unsigned short int	0 bis 65 535
unsigned long int	0 bis 4 294 967 295
char	alle ASCII-Zeichen
unsigned char	alle ASCII-Zeichen
float	-10^{38} bis 10^{38}
double	-10^{308} bis 10^{308}

Früher gab es zwei Arten ASCII-Zeichen, (das sind die darstellbaren Zeichen), zu kodieren. Der erste Wertebereich von -128 bis 127, der zweite Wertebereich von 0 bis 255.

Um diesem Problem der Unterschiedlichen ASCII-Werte zu entgehen, wurde der *unsigned char* Typ eingeführt. Dieser Typ kennt nur Werte von 0 bis 255.

Prinzipiell wird heute nur noch mit *unsigned char* gearbeitet

Parameter für printf

In unseren Beispielen geben wir die Ergebnisse mit *printf* im Diagnosefenster von *Global Script* aus. Hierzu einige Informationen zu *printf*.

Aufbau und Beispiel für *printf*:

```
printf("Inhalt der Variable: %d\r\n",variable);
```

Das % ist ein Sonderzeichen, genau wie der \. Das % sagt dem Programm, daß dort der Wert einer Variable stehen soll. Im Fall %d wird eine *integer-Variable* angezeigt.

Aufzählungen von Variablen werden durch Kommas getrennt.


Hier ein kleiner Auszug der Parameter von *printf*:

Parameter	Bedeutung
%d	int, short int oder char (als Dezimalzahl)
%ld	long int als Zahl darstellen
%c	wie %d mit char als Zeichen
%x	wie %d mit char als Hexadezimal mit kleinem a...f
%X	wie %x nur mit großen Hexadezimalzeichen A...F
%o	wie %d mit char als Oktalzeichen
%u	wie %d , aber nur unsigned Werte
%f	Floatzahlen in Fließkommenschreibweise , z.B. 3.43234
%e	Floatzahlen in Exponentialschreibweise , z.B. 23e+432
%E	wie %e nur mit einem großen E , z.B. 23E+432
%g	Floatzahl in Exponential- oder Fließkommenschreibweise
%s	char* bzw. char[]
%le	Double-Zahl darstellen
%%	ist keine Anweisung, gibt aber ein % aus
\n	ist keine Anweisung, gibt aber ein Carige Return (Zeilenwechsel) aus
\r	ist keine Anweisung, gibt aber ein Line feed (Zeilenvorschub) aus
\t	ist keine Anweisung, gibt aber einen Tabulator aus

4.2.1 Beispiel 1 - Variablentypen verwenden

In unserem ersten Beispiel zeigen wir Ihnen, wie man die unterschiedlichen Variablentypen in einer *C-Aktion* verwendet.

Die Aktion ist am Button *example_01* → *Ereignis* → *Maus* → *links drücken* projiziert.

Beim Drücken des Button *Beispiel 1* mit der  wird das nachfolgende Script bearbeitet.

C-Aktion

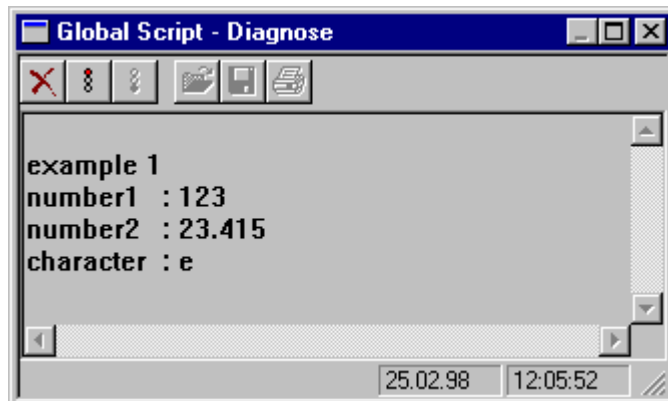
```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                  char* lpszPropertyName, UINT nFlags, int x, int y)
{
//tag declaration
int number1;
float number2 = 23.415;//set value in declaration
char character;

//set tag values
number1 = 123;
character = 'e';

//output in diagnostics window
printf("\r\nexample 1\r\n");
printf("number1\t : %d\r\n",number1);
printf("number2\t : %.3f\r\n",number2);
printf("character\t : %c\r\n",character);

//set internal tag
SetTagDWord("S32i_course_tag_03",number1);
}
```

Ausgabe im Diagnosefenster:




Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Im Abschnitt *Deklaration* werden die Variablen (*number1*, *number2* und *character*) erzeugt. Dies geschieht durch die Schreibweise *Variablentyp Variablenname* und jede Zeile wird mit ; abgeschlossen.
- Bei der Variable *number2* erfolgt die Wertzuweisung mit der Deklaration. Dies geschieht durch die Schreibweise *Variablentyp Variablenname = Wert*, der zugewiesen werden soll.
- Den restlichen Variablen werden die Werte zugewiesen. Wir haben nun die *Variablen* mit einer bestimmten Größe (*int*, *float* und *char*).
- Die *Variablen* haben nun die Werte (*123*, *23.415*, und *e*). Die Trennung von Vorkommazahl und Nachkommazahl erfolgt durch einen Punkt.
- Anschließend werden die Werte mit der Funktion *printf* im Diagnosefenster ausgegeben.

4.2.2 Beispiel 2 - C Variablen in Verbindung mit WinCC Variablen

In unserem zweiten Beispiel zeigen wir Ihnen die Verwendung von C Variablen in Verbindung mit WinCC Variablen, hier *interne Variablen*. Die *externen Variablen* (Prozeßvariablen aus dem Variablenhaushalt) werden auf die gleiche Art und Weise gelesen bzw. verändert.

Dieses Beispiel zeigt das Setzen einer (internen oder externen) Variable auf einen definierten Wert (*SetTag*) als auch das Lesen des aktuellen Wertes (*GetTag*) einer Variable. Die Aktion ist am Button *example_02* → *Ereignis* → *Maus* → *links drücken* projiziert.

Beim Drücken des Button *Beispiel 2* mit der  wird das nachfolgende Script bearbeitet.

C-Aktion

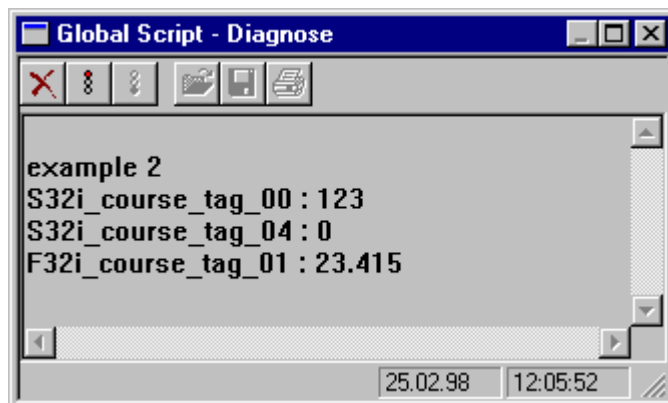
```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                  char* lpszPropertyName, UINT nFlags, int x, int y)
{
    //tag declaration
    int    number1;
    int    number2;
    float  number3;

    //set internal tags
    SetTagDWord("S32i_course_tag_00",123);
    SetTagFloat("F32i_course_tag_01",23.415);

    //read internal tag values
    number1=GetTagDWord("S32i_course_tag_00");
    number2=GetTagDWord("S32i_course_tag_04");
    number3=GetTagFloat("F32i_course_tag_01");

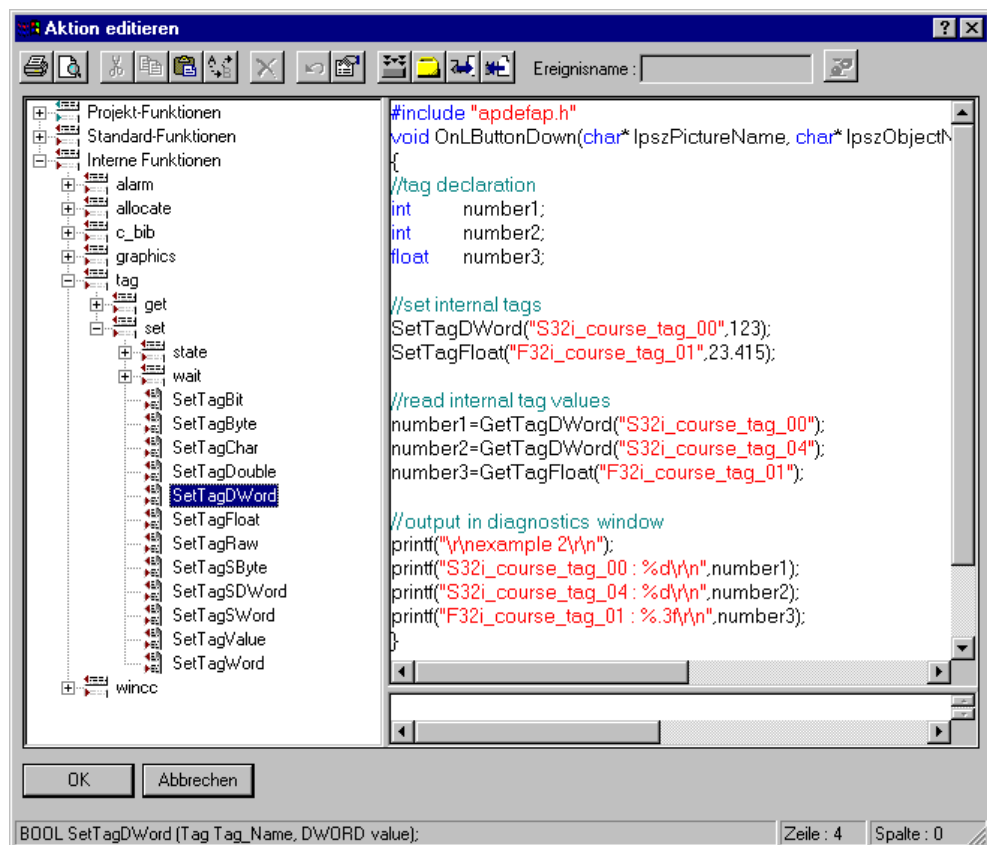
    //output in diagnostics window
    printf("\r\nexample 2\r\n");
    printf("S32i_course_tag_00 : %d\r\n",number1);
    printf("S32i_course_tag_04 : %d\r\n",number2);
    printf("F32i_course_tag_01 : %.3f\r\n",number3);
}
```


Ausgabe im Diagnosefenster:



Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der C Variablen.
- Im Abschnitt *Set internal tags* werden die internen WinCC Variablen (*S32i_course_tag_00* und *F32i_course_tag_01*) mit den dafür vorgesehen WinCC Funktionen *SetTagDWord* und *GetTagFloat* mit Werten (123 und 23.415) versorgt.




- Im folgenden Abschnitt lesen wir mit der entsprechenden WinCC Funktion (*GetTagDWord*, *GetTagFloat*) den Wert aus den internen WinCC Variablen (*S32i_course_tag_00*, *S32i_course_tag_01*) und weisen sie den C Variablen (*number1*, *number3*) zu.
- Der Wert der Variable (*S32i_course_tag_04*) wird aus dem Eingabefeld 1 gelesen und der C Variable (*number2*) zugewiesen. Dieser Wert kann geändert werden. Klicken Sie mit der  in das *Eingabefeld 1*, geben den Wert ein und beenden Sie die Eingabe mit *Return*.
- Anschließend werden die Werte mit der Funktion *printf* im Diagnosefenster ausgegeben.

4.2.3 Beispiel 3 - Verwendung von Variablen

In unserem dritten Beispiel zeigen wir Ihnen eine weitere Möglichkeit der Verwendung einer Variable. Wir ändern bei Mausklick den Inhalt einer Variable. Diese Variable ist mit der X-Position eines Objektes verbunden. Bei Änderung der Variable wird das Objekt verschoben.

Die Aktion ist am Button *example_03* → Ereignis → Maus → links drücken projiziert.

Beim Drücken des Button *Beispiel 3* mit der  wird das nachfolgende Script bearbeitet.

C-Aktion

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                  char* lpszPropertyName, UINT nFlags, int x, int y)
{
    int number;

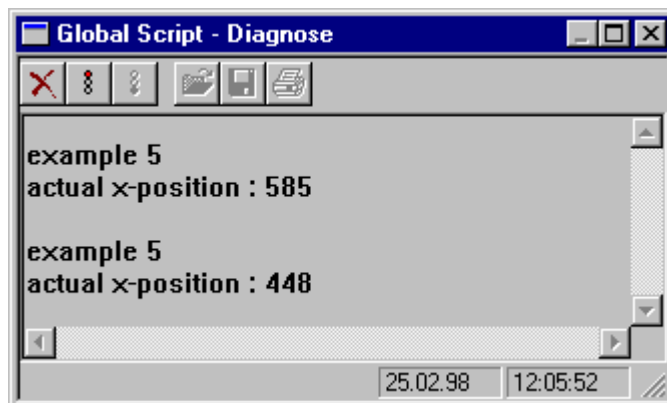
    number=GetTagDWord("S32i_course_tag_02");

    //set tags
    if (number==448) (number=585);
    else (number=448);

    SetTagDWord("S32i_course_tag_02",number);

    //output in diagnostics window
    printf("\r\nexample 5\r\n");
    printf("actual x-position : %d\r\n",number);
}
```

Ausgabe im Diagnosefenster:



Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der Variable.
- Im Abschnitt *set tags* wird in der *if-Bedingung* der Wert der Variable getauscht.
- Der neue Wert wird der internen WinCC Variablen (*S32i_course_tag_02*) mit der dafür vorgesehenen WinCC Funktion *SetTagDWord* zugewiesen.
- Beim Objekt *rectangle_01* ist diese Variable mit der *PositionX* verknüpft. Die Aktualisierung der Position erfolgt bei Änderung dieser Variable *S32i_course_tag_02*.
- Anschließend wird der Wert mit der Funktion *printf* im Diagnosefenster ausgegeben.

4.3 Operatoren und mathematische Funktionen in C




In unserem Projekt erreichen Sie die Beispiele zum Thema Operatoren durch die Anwahl des Button *Operatoren*. Mit der  wird der Sourcecode zum jeweiligen Beispiel angezeigt.

Bild für Operatoren



Nach der Anwahl des Button *Operatoren* mit der  sehen Sie das oben dargestellte Bild. Die einzelnen Button im Bild führen die beschriebenen Beispiele aus. Mit der  wird der Sourcecode zum Beispiel angezeigt.

Bevor wir mit den Beispielen beginnen, etwas Theorie zum Thema Operatoren. Die *Operatoren* kontrollieren was mit den *Variablen* und *Konstanten* geschehen soll. Man verwendet *Operatoren* um *Variablen* und *Konstanten* miteinander zu verknüpfen und daraus neue Inhalte von *Variablen* zu erhalten.

Operatoren werden in unterschiedliche Typen eingeteilt. Nachfolgend sind die für uns wichtigsten:

Arithmetik

Typ	Beschreibung
+	Addition
-	Subtraktion
*	Multiplikation
/	Division
++	inkrementieren
--	dekrementieren

Die arithmetischen Operatoren werden z.B. im *Dynamik-Dialog* für die Abhängigkeit mehrerer Variablen benötigt (z.B. Motor-Status1 und Motor-Status2). Die arithmetischen Operatoren werden außerdem für die Formulierung von Formeln innerhalb von *C-Aktionen* eingesetzt.

Logik, Vergleich

Typ	Beschreibung
>	größer
>=	größer gleich
==	gleich
!=	ungleich
<=	kleiner gleich
<	kleiner
&&	logische UND Verknüpfung
	logische ODER Verknüpfung
!	logische Umkehrung

Die arithmetischen und logischen Operatoren werden z.B. im *Dynamik-Dialog* für die Abhängigkeit mehrerer Variablen benötigt (z.B. Motor-Status und !Störung).

Die logischen Operatoren finden sich in den Abfragen von *C-Aktionen* wieder.

Bit Manipulation

Typ	Beschreibung
&	UND Verknüpfung von Bits
	ODER Verknüpfung von Bits
^	Exklusiv ODER Verknüpfung von Bits
<<	Bit Verschiebung nach links
>>	Bit Verschiebung nach rechts

Die bitweisen Operatoren werden z.B. in *C-Aktionen* eingesetzt, um Bits abzufragen oder einzelne Bits in einem Datenwort zu setzen (z.B. Motor-Status-Bit im Wort an Position 1 setzen, Motorstatus | 0x0002).

4.3.1 Boolesche Algebra

Logische Funktionen

In der Mathematik werden Systeme logischer Variablen, die über logische Funktionen verknüpft sind, als **Boolesche Algebra** bezeichnet.

Bei der Erstellung von *C-Aktionen* werden in vielen Anwendungsfällen logische Variablen über logische Funktionen zu neuen Variablen verknüpft.

Eine logische Funktion lässt sich in einer **Funktionstabelle** darstellen. Eine Funktionstabelle kann man auch als **Wahrheitstabelle** bezeichnen.

Will man logische Verknüpfungsschaltungen darstellen, so verwendet man spezielle Symbole, die standardisiert sind. Man kann diese Symbole auch als **Gatter** bezeichnen.


4.3.1.1 Logische Grundfunktionen

Anschließend werden die wichtigsten logischen Grundfunktionen dargestellt. Logische Schaltungen werden vielerorts mit ihren englischen Namen angegeben.

Inverterer

Der **Inverterer** negiert das Eingangssignal. Die Negation wird über den Kreis am Ausgang der Box dargestellt.


a	Not
0	1
1	0



Und- Verknüpfung

Die **Und- Verknüpfung** ist nur dann eins, wenn beide Variablen den Wert eins annehmen.

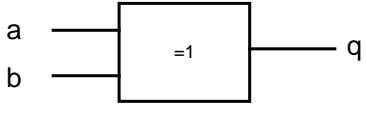
a	b	And
0	0	0
0	1	0
1	0	0
1	1	1



Oder- Verknüpfung

Die **Oder- Funktion** ist dann eins, wenn der Wert mindestens einer Variablen eins ist.


a	b	Or
0	0	0
0	1	1
1	0	1
1	1	1



Nicht- Und- Verknüpfung

Eine Und-Verknüpfung gefolgt von einer Negation bezeichnet man als **NAND-Verknüpfung**. Diese Funktion hat den Wert Null, wenn beide Variablen den Wert eins annehmen.

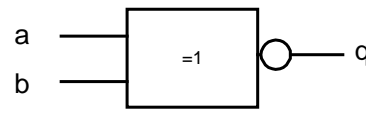
a	b	Nand
0	0	1
0	1	1
1	0	1
1	1	0



Nicht- Oder- Verknüpfung

Eine Oder- Verknüpfung gefolgt von einer Negation bezeichnet man als **Nor-Verknüpfung**. Diese Funktion hat nur dann den Wert eins, wenn beide Variablen den Wert Null haben.


a	b	Nor
0	0	1
0	1	0
1	0	0
1	1	0



Exklusiv- Oder- Verknüpfung

Diese Funktion ergibt dann den Wert eins, wenn entweder die eine oder die zweite Variable den Wert eins annimmt. Diese Funktion bezeichnet man als **EXOR- Funktion**.


a	b	EXOR
0	0	0
0	1	1
1	0	1
1	1	0



4.3.2 Beispiel 1 - Operatoren der Grundrechnungsarten verwenden

In Beispiel 1 zeigen wir Ihnen, wie man die unterschiedlichen Operatoren der Grundrechnungsarten in einer *C-Aktion* verwendet.

Die Aktion ist am Button *example_01* → *Ereignis* → *Maus* → *links drücken* projiziert.

Beim Drücken des Button *Beispiel 1* mit der  wird das nachfolgende Script bearbeitet.

C-Aktion

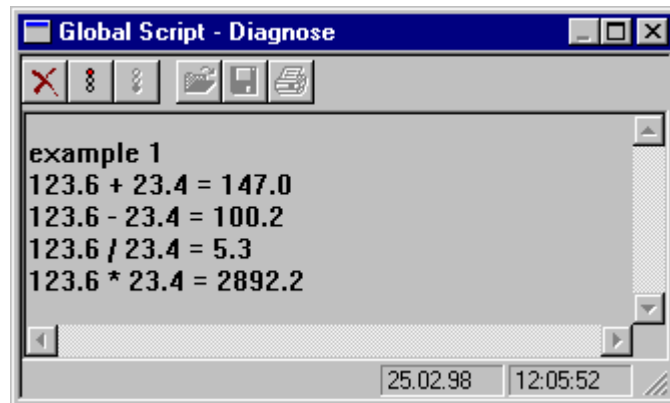
```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                  char* lpszPropertyName, UINT nFlags, int x, int y)
{
float a,b;
float result1,result2,result3,result4;

a=123.6;
b=23.4;

result1=a+b;    // add
result2=a-b;    // subtract
result3=a/b;    // divide
result4=a*b;    // multiply

//output in diagnostics window
printf("\r\nexample 1\r\n");
printf("%.1f + %.1f = %.1f\r\n",a,b,result1);
printf("%.1f - %.1f = %.1f\r\n",a,b,result2);
printf("%.1f / %.1f = %.1f\r\n",a,b,result3);
printf("%.1f * %.1f = %.1f\r\n",a,b,result4);
}
```


Ausgabe im Diagnosefenster:



Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der C Variablen.
- Den Variablen werden die Werte zugewiesen.
- Ausführen der mathematischen Operationen.
- Anschließend werden die Werte mit der Funktion *printf* im Diagnosefenster ausgegeben.

4.3.3 Beispiel 2 - mathematische Funktionen

In Beispiel 2 zeigen wir wie man mathematische Funktionen in einer *C-Aktion* verwendet. Die Aktion ist am Button *example_02* → *Ereignis* → *Maus* → *links drücken* projiziert. Beim Drücken des Button *Beispiel 2* mit der  wird das nachfolgende Script bearbeitet.

C-Aktion

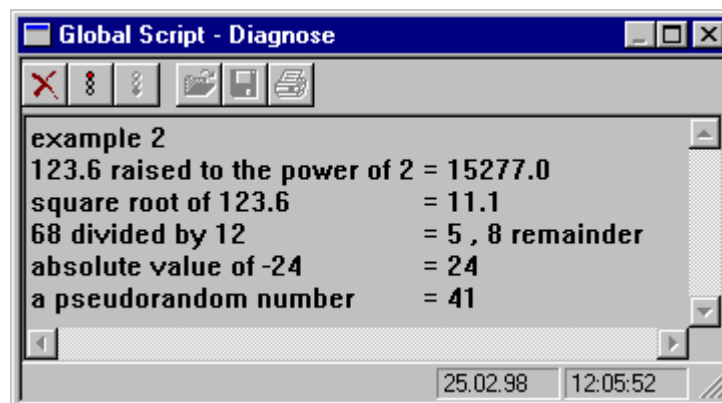
```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                  char* lpszPropertyName, UINT nFlags, int x, int y)
{
float a;
int b,c,d;
div_t result5;
double result1,result2;
int result3,result4;

a=123.6;
b=-24;
c=68;
d=12;

result1=pow(a,2);
result2=sqrt(a);
result3=abs(b);
result4=rand();
result5=div(c,d);

//output in diagnostics window
printf("\r\nexample 2\r\n");
printf("%.1f raised to the power of 2\t = %.1f\r\n",a,result1);
printf("square root of %.1f\t = %.1f\r\n",a,result2);
printf("%d divided by %d\t\t = %d , %d
remainder\r\n",c,d,result5.quot,result5.rem);
printf("absolute value of %d\t = %d\r\n",b,result3);
printf("a pseudorandom number\t = %d\r\n",result4);
}
```

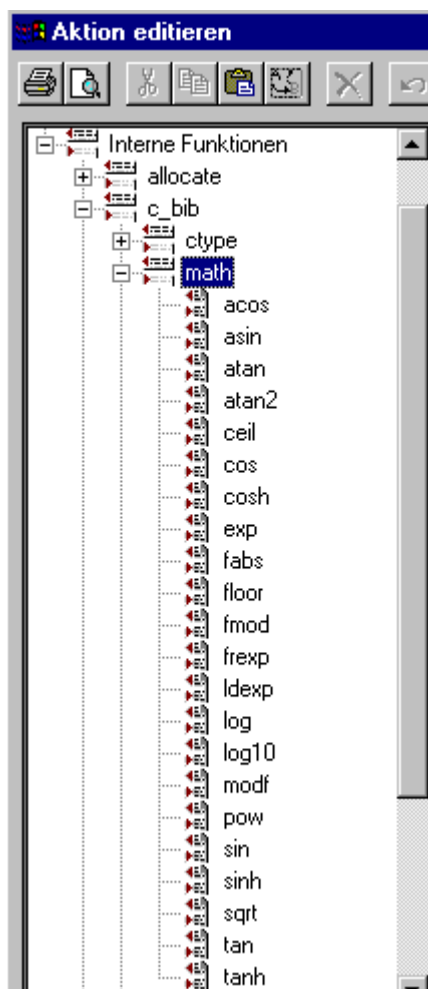
Ausgabe im Diagnosefenster:



Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der C Variablen.
- Der Variablen werden die jeweiligen Werte zugewiesen.
- Ausführen der mathematischen Funktionen. Sie finden diese Funktionen bei → *internen Funktionen* → *c_bib* → *math*
- Anschließend werden die Werte mit der Funktion *printf* im Diagnosefenster ausgegeben.


Immer wenn mathematische Funktionen anwählbar sind, können sie eingesetzt werden. Zum Beispiel im *Dynamik-Dialog* bei der Formulierung der Abfrage oder in den *C-Aktionen*.



4.3.4 Beispiel 3 - Operatoren für Bitoperationen

In Beispiel 3 zeigen wir Ihnen, wie man Operatoren für Bitoperationen in einer *C-Aktion* verwendet.

Die Aktion ist am Button *example_03* → *Ereignis* → *Maus* → *links drücken* projiziert.

Beim Drücken des Button *Beispiel 3* mit der  wird das nachfolgende Script bearbeitet.

C-Aktion

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                  char* lpszPropertyName, UINT nFlags, int x, int y)
{
    int a,b,c;
    int result1,result2,result3,result4,result5;

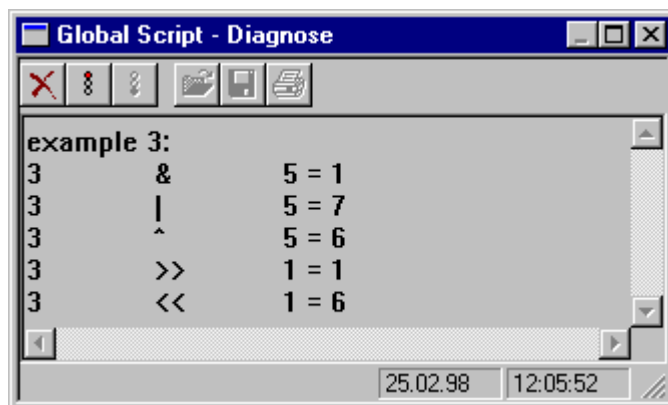
    a=3; // binary 011
    b=5; // binary 101

    result1=a&b; // AND
    result2=a|b; // inclusive OR
    result3=a^b; // exclusive OR
    result4=a>>1; // shift right
    result5=a<<1; // shift left

    //write result of OR in output_01
    SetTagDWord("S32i_course_op_03",result2);

    //output in diagnostic window
    printf("\r\nexample 3:\r\n");
    printf("%d\t&\t%d = %d\r\n",a,b,result1);
    printf("%d\t|\t%d = %d\r\n",a,b,result2);
    printf("%d\t^\t%d = %d\r\n",a,b,result3);
    printf("%d\t>>\t1 = %d\r\n",a,result4);
    printf("%d\t<<\t1 = %d\r\n",a,result5);
}
```

Ausgabe im Diagnosefenster:



Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der C Variablen und die Zuweisung der Werte.
- Ausführen der Bitoperationen.
- Ausgabe des Ergebnisses der *inclusive ODER* Verknüpfung im *Ausgabefeld 1*.
- Anschließend werden die Werte mit der Funktion *printf* im Diagnosefenster ausgegeben.

4.4 Zeiger in C




In unserem Projekt erreichen Sie die Beispiele zum Thema Zeiger durch die Anwahl des Button *Zeiger*. Mit der  wird der Sourcecode zum jeweiligen Beispiel angezeigt.

Bild für Zeiger



Nach der Anwahl des Button *Zeiger* mit der  sehen Sie das oben dargestellte Bild. Die einzelnen Button im Bild führen die beschriebenen Beispiele aus. Mit der  wird der Sourcecode zum Beispiel angezeigt.

Bevor wir mit den Beispielen beginnen, etwas Theorie zum Thema Zeiger.

Die Zeiger sind ein wesentlicher Bestandteil der Sprachen C und C++. Der Zeiger wird auch in deutsch oft als **Pointer** bezeichnet.

Die Definition und Schreibweise von Zeigern

Ein Zeiger beinhaltet nicht den Wert einer Variable, sondern **zeigt** nur **auf die Adresse**, wo der Wert der Variablen hinterlegt ist.

Der Zeiger muß den selben Datentyp besitzen wie die Variable auf die er zeigt!

Grundsätzlich wird eine Variable durch einen **vorangestellten Stern** als Zeiger deklariert.

Der Zugriff auf den Variablenwert, auf den ein Zeiger zeigt, erfolgt mittels des

*** Operators**. Man bezeichnet ihn auch als Inhalts-Operator.

Dieser ist jedoch nicht mit dem Multiplikator zu verwechseln. Das sind zwei unterschiedliche Operatoren!

Der Compiler entscheidet aus dem Zusammenhang, ob es sich um einen Zeiger-Operator oder um eine Multiplikation handelt.

Programmauszug zur Erklärung der Zeiger

```
{
int number1;
int *pointer1;

number1 = 123;
pointer1 = &number1;
}
```

Anhand des Beispiels wird am besten verdeutlicht, was ein Zeiger ist:

- Wir erzeugen die Variable *number1* und den Zeiger *pointer1*.
- Der Variable *number1* wird der Wert *123* zugewiesen.
- Dem Zeiger *pointer1* wird **die Adresse** der Variable *number1* zugewiesen.
- Dies geschieht durch den *Adressoperator &!*
- Nun steht im *pointer1* die Adresse der Variable *number1*.

Der Vorteil liegt nun darin, daß in den Zeigern nur eine Adresse und keine Variablen stehen. Dadurch sind sie extrem flexibel und schnell.

String als Zeiger

Wird ein String (Zeichenfolge) durch einen Zeiger erzeugt, so wird der String im Speicher an einer beliebigen freien Stelle angelegt. Der Zeiger zeigt auf das erste Element des Strings.

Es funktioniert zwar, ist aber **ein schlechter und gefährlicher Programmierstil!**

Eine Folge von Zeichen – als String bezeichnet – wird in C durch den Anfang der Zeichenfolge (Startadresse der Zeichenfolge) und durch das Ende (Nullzeichen, \0) gekennzeichnet:

W	i	n	C	C	\0
---	---	---	---	---	----

In der WinCC-Projektierung finden Sie Zeiger z.B. in Zusammenhang mit den Funktionsparametern wieder:

IpszPictureName ist zum Beispiel ein Zeiger auf den Anfang des Namens des Bildes (z.B. Start),

S	t	a	r	t	\0
---	---	---	---	---	----

IpszObjectName ist ein Zeiger auf den Anfang des Objektnamens (z.B. Kreis1) .

K	r	e	i	s	1	\0
---	---	---	---	---	---	----


Der wahren Nutzen von Zeigern wird erst bei ihrer Verwendung in **Feldern deutlich**. Diese werden in deutsch oft als **Arrays** bezeichnet.

Gruppen oder Aneinanderreihungen von Variablen eines Typs, kann man als Felder bezeichnen.

4.4.1 Beispiel 1 - Zeiger

In Beispiel 1 zeigen wir Ihnen die Verwendung von Zeigern in C.

Die Aktion ist am Button *example_01* → *Ereignis* → *Maus* → *links drücken* projiziert.

Beim Drücken des Button *Beispiel 1* mit der  wird das nachfolgende Script bearbeitet.

C-Aktion

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                  char* lpszPropertyName, UINT nFlags, int x, int y)
{
//declaration of tags
int number1;
float number2;
char character;

//declaration of pointers
int *pointer1;
float *pointer2;
char *pointerc;

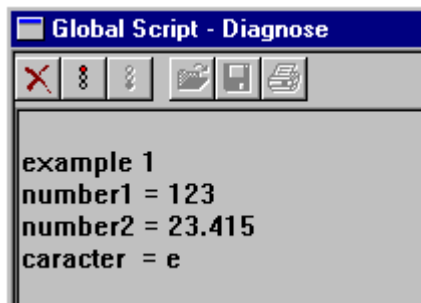
//set tag values
number1 =123;
number2 =23.415;
character='e';

//set addresses
pointer1=&number1;
pointer2=&number2;
pointerc=&character;

//set internal tag
SetTagDWord("S32i_course_point_00",*pointer1);

//output in diagnostics window
printf("\r\nexample 1\r\n");
printf("number1 = %d\r\n",GetTagDWord("S32i_course_point_00"));
printf("number2 = %.3f\r\n",*pointer2);
printf("character = %c\r\n",*pointerc);
}
```

Ausgabe im Diagnosefenster:




Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der C Variablen.
- Die Deklaration der Zeiger.
- Den Variablen Werte zuweisen.
- In diesem Abschnitt werden den *Zeigern* die *Adressen* der entsprechenden Variablen zugewiesen
- Anschließend werden die Werte mit der Funktion *printf* im Diagnosefenster ausgegeben.

4.4.2 Beispiel 2 - Zeiger in Verbindung mit WinCC Variablen

In Beispiel 2 zeigen wir Ihnen die Verwendung von Zeigern im Zusammenhang mit *internen Variablen*.

Die Aktion ist am Button *example_02* → Ereignis → Maus → links drücken projiziert.

Beim Drücken des Button *Beispiel 2* mit der  wird das nachfolgende Script bearbeitet.

C-Aktion

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                  char* lpszPropertyName, UINT nFlags, int x, int y)
{
//declaration of array
int array[4];

int index;
int *pointer;

//set tag values
array[0]=2;
array[1]=4;
array[2]=6;
array[3]=8;

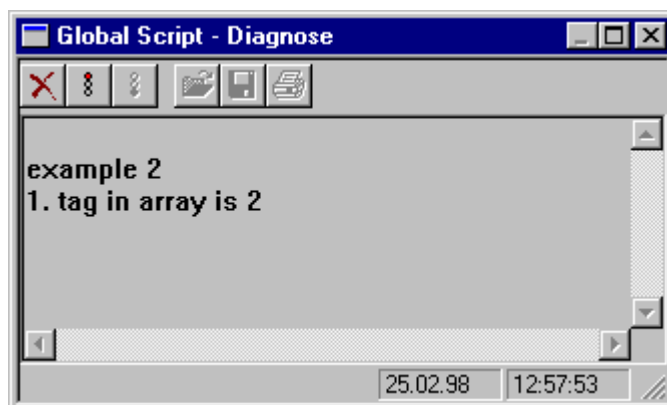
//pointer to first tag of array
pointer=&array[0];

//read index from input_01
index=GetTagDWord("S32i_course_point_04");
index--;

//set internal tag
SetTagDWord("S32i_course_point_03", *(pointer+index));

//output in diagnostics window
printf("\r\nexample 2\r\n");
printf("%d. tag in array is %d\r\n", (index+1), *(pointer+index));
}
```


Ausgabe im Diagnosefenster:



Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration eines Feldes (vier integer Werte).
- Die Deklaration einer Variablen.
- Die Deklaration eines Zeiger.
- Dem Feld Werte zuweisen
- In diesem Abschnitt wird dem Zeiger die Adresse des ersten Elementes des Feldes zugewiesen
- Den gewünschten Index aus dem Eingabefeld 1auslesen und auf das C Format korrigieren (das erste Element in einem Feld hat den Index 0).
- Zum Zeiger auf die erste Position im Feld wird der aktuelle Index addiert. Mit dem so berechneten Zeiger wird das gewünschte Element ausgelesen.
- Anschließend werden die Werte mit der Funktion *printf* im Diagnosefenster ausgegeben.

4.4.3 Beispiel 3 - Zeiger in Verbindung mit Stringverarbeitung

In Beispiel 3 zeigen wir Ihnen die Verwendung von Zeigern in C. Wir ändern die Darstellung eines Textes in Form einer Laufleiste
 Die Aktion ist am Statischen Text *static text_01* → *Eigenschaften* → *Schrift* → *Text* projiziert. Beim Drücken des Button *Beispiel 3* mit der  wird durch eine Direktverbindung die Variable *BINi_varia_point_05* auf 1 gesetzt. Das nachfolgende Script wird alle 250ms bearbeitet. Die if-Bedingung wird aber nur bei gedrücktem Button *Beispiel 3* ausgeführt.

C-Aktion

```
#include "apdefap.h"
long _main(char* lpszPictureName, char* lpszObjectName, char*
           lpszPropertyName)
{
//declaration of static tag
static int i = 10;

char *str = "          WinCC";

if (GetTagBit("BINi_varia_point_05")) {
    if (i>19) (i=0); //check limit
    i++; //inc
}
//return string
return(str+i);
}
```

Die Auswirkungen sind direkt im Bild sichtbar:



Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der Variable vom Typ *static in*. Variablen mit vorangestelltem *static* behalten Ihren Wert für die Dauer der Bildanwahl. Die Initialisierung erfolgt nur beim Ersten Aufruf der Aktion.
- Die Deklaration des Zeigers und die Wertzuweisung.
- Überprüfen ob der Button *Beispiel 3* gedrückt ist.
- Überprüfen ob die Position die maximale Zeichenlänge verläßt. Wenn dies der Fall ist, wird wieder von vorne begonnen.
- Die Position incrementieren.
- Es wird der Zeiger auf die Zeichenfolge um die Position erhöht und das Ergebnis mit *return* zurückgegeben.

4.5 Schleifen und bedingte Anweisungen in C




In unserem Projekt erreichen Sie die Beispiele zum Thema Schleifen durch die Anwahl des Buttons *Schleifen*. Mit der  wird der Sourcecode zum jeweiligen Beispiel angezeigt.

Bild für Schleifen



Nach der Anwahl des Button *Schleifen* mit der  sehen Sie das oben dargestellte Bild. Die einzelnen Button im Bild führen die beschriebenen Beispiele aus. Mit der  wird der Sourcecode zum Beispiel angezeigt.

Ü und 3 sind weitere Beispiele für Schleifen.

Schleifen

Bevor wir mit den Beispielen beginnen, etwas Theorie zum Thema Schleifen.

Prinzipiell gibt es zwei Arten von Schleifen: Die *vorprüfenden* und die *nachprüfenden* Schleifen.

Die *vorprüfenden Schleifen* prüfen **vor** dem Schleifenkörper und die *nachprüfenden Schleifen* **nach** dem Schleifenkörper, ob eine Bedingung wahr ist. Das heißt, nachprüfende Schleifen werden mindestens 1x durchlaufen.

Wenn der Schleifenkörper aus einem Befehl besteht, können die geschweiften Klammern weggelassen werden.

while

```
while ([Bedingung])
{ Schleifenkörper }
```

Der Schleifenkörper wird wiederholt, solange die *Bedingung* erfüllt ist.

do - while

```
do
{ Schleifenkörper }
while [Bedingung] ;
```

Der Schleifenkörper wird mindestens 1x durchlaufen und anschließend solange wiederholt, bis die Bedingung nicht mehr erfüllt ist.

Hinweis:

Hinter der Anweisung while-Bedingung steht bei den *nachprüfenden Schleifen* **immer** ein Semikolon; bei den *vorprüfenden Schleifen* dagegen darf **kein** Semikolon stehen.

for

```
for ( [ Init ] ; [ Bedingung ] ; [ Anweisung ] )
    { Schleifenkörper }
```

Sieht alles sehr kryptisch aus. Also schreiben wir das ganze etwas um und erhalten als equivalent folgenden Ausdruck:

```
Init;
while [Bedingung]
    {Schleifenkörper;
    [Anweisung];
    }
```

Hinweis:

Beachten Sie, daß Schleifen auch zu Endlosschleifen werden können, wenn die Bedingung **immer** erfüllt wird. Aus diesem Grunde testen Sie Ihre *C-Aktionen* sofort nach der Formulierung, ob die Sequenz fehlerfrei ist.

Bedingte Anweisungen

Bei Schleifen wird ein Schleifenkörper bearbeitet, solange die Bedingung wahr ist. In bedingten Anweisungen wird die Anweisung, wenn die Bedingung wahr ist, genau einmal bearbeitet. Bei Vergleichen immer auf die Variablentypen achten!

if

```
if ( [Bedingung1] )
    { Körper1 }
else
    { Körper2 }
```

Wenn die Bedingung wahr (ungleich 0) ist, wird der Befehls-Körper1 bearbeitet. Wenn die Bedingung nicht zutrifft (der Wert ist 0), wird die Alternative im Befehls-Körper2 bearbeitet. *else* kann man auch weglassen, dann wird keine Alternative ausgeführt.

Da bei *if* einfach der numerische Wert eines Ausdrucks überprüft wird, sind bestimmte Abkürzungen möglich. Die beiden nachfolgenden Anweisungen sind daher identisch.

```
if (Bedingung)
if (Bedingung != 0)
```

Man kann mehrere Abfragen kombinieren:

```
if ( [Bedingung1] )
    { Körper1 }
else if ( [Bedingung2] ) // zweite if Abfrage (Verschachtelt)
    { Körper2 }
else
    { Körper3 }
```

Hier wird erst die erste Bedingung geprüft. Trifft diese nicht zu, wird die zweite geprüft. Wenn ja, so wird der Befehls-Körper2 ausgeführt. Trifft keine der Bedingungen zu, so wird der Befehls-Körper3 ausgeführt.

Hinweis:

Wird eine Bedingung ausgeführt, so werden die anderen Fälle nicht mehr bearbeitet.

Bei mehr als zwei Abfragen ist es besser switch case zu verwenden.

switch & case


```
switch ( [Variable] ) {
    case [Term1] :
        Aktion1;
        break;
    case [Term2] :
        Aktion2;
        break;
    default:
        Aktion3;
        break;
}
```

Hier wird eine Variable auf Gleichheit untersucht. Mit der *switch*-Anweisung gibt man die zu untersuchende Variable an. Zuerst wird geprüft, ob die Variable dem *Term1* entspricht, wenn ja, so wird *Aktion1* ausgeführt. Diese Aktion kann beliebig viele Befehle beinhalten. Abgeschlossen werden die Aktionen mit einem *break*!

Hinter dem Schlüsselwort *default* steht kein Vergleich. Diese Aktionen werden erst dann ausgeführt, wenn keine der *case* Bedingungen erfüllt ist. Also eine Default-Anweisung, solange kein anderer Fall eintritt.

Die *break*-Anweisung kann bei *default* (ist immer die letzte Anweisung) entfallen, es ist aber schlechter Programmierstil.

4.5.1 Beispiel 1 - while - Schleife

In Beispiel 1 zeigen wir die Verwendung der *while* Schleife in einer *C-Aktion*.
Die Aktion ist am Button *example_01* → *Ereignis* → *Maus* → *links drücken* projiziert.
Beim Drücken des Button *Beispiel 1* mit der  wird das nachfolgende Script bearbeitet.

C-Aktion

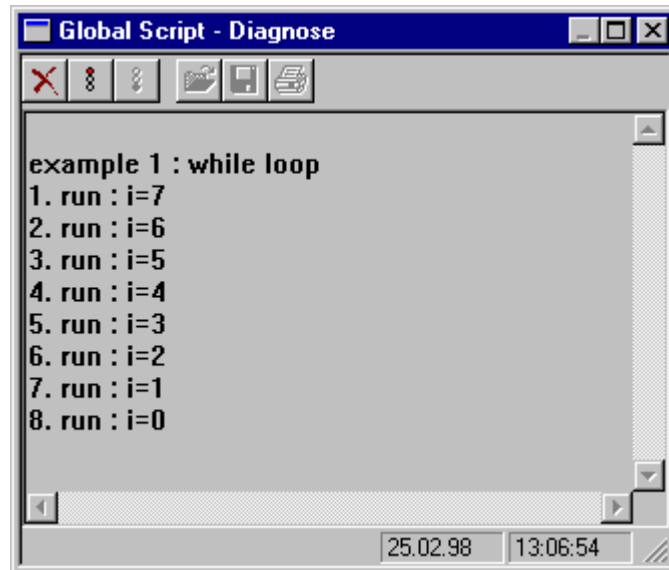
```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                  char* lpszPropertyName, UINT nFlags, int x, int y)
{
//while loop
int i,j;

printf("\r\nexample 1 : while loop\r\n");

j=0; // execution count
i=8; // tag used for loop action

while (i>0) {
    j++;
    i--;
    printf("%d. run : i=%d\r\n",j,i);
} //while
}
```


Ausgabe im Diagnosefenster:



Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der C Variablen. *i* wird für die Schleifenfunktion benötigt, *j* nur für die Ausgabe der Durchläufe.
- Im Diagnosefenster ausgeben, was anschließend kommt.
- Die Variablen für die Schleifenbearbeitung initialisieren.
- Die Bearbeitung der *while* Schleife solange $i > 0$ ist.
- Die Variable *j* wird inkrementiert, *i* wird dekrementiert.
- Anschließend wird der aktuelle Wert der Variablen mit der Funktion *printf* im Diagnosefenster ausgegeben.

4.5.2 Beispiel 2 - do - while - Schleife

In Beispiel 2 zeigen wir Ihnen die Verwendung der *do - while* Schleife in einer *C-Aktion*. Die Aktion ist am Button *example_02* → *Ereignis* → *Maus* → *links drücken* projiziert. Beim Drücken des Button *Beispiel 2* mit der  wird das nachfolgende Script bearbeitet.

C-Aktion

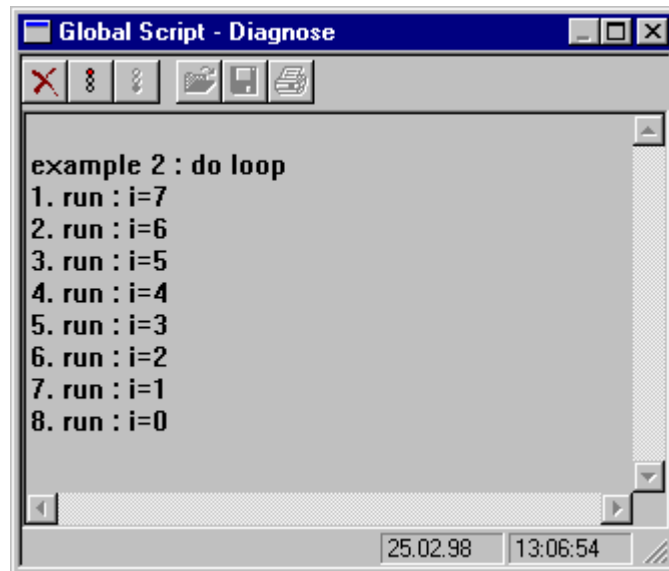
```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                  char* lpszPropertyName, UINT nFlags, int x, int y)
{
//do loop
int i,j;

printf("\r\nexample 2 : do loop\r\n");

j=0; //execution count
i=8; //tag used in loop action

do {
    j++;
    i--;
    printf("%d. run : i=%d\r\n",j,i);
} //do
while (i>0);
}
```


Ausgabe im Diagnosefenster:



Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der C Variablen. *i* wird für die Schleifenfunktion benötigt, *j* nur für die Ausgabe der Durchläufe.
- Im Diagnosefenster ausgeben, was anschließend kommt.
- Die Variablen für die Schleifenbearbeitung initialisieren.
- Die Bearbeitung der *do - while* Schleife erfolgt solange $i > 0$ ist (mindestens 1 Durchlauf).
- Die Variable *j* wird inkrementiert, *i* wird dekrementiert.
- Anschließend wird der aktuelle Wert der Variablen mit der Funktion *printf* im Diagnosefenster ausgegeben.

4.5.3 Beispiel 3 - for Schleife

In Beispiel 3 zeigen wir Ihnen die Verwendung der *for* Schleife in einer *C-Aktion*. Die Aktion ist am Button *example_03* → *Ereignis* → *Maus* → *links drücken* projiziert. Beim Drücken des Button *Beispiel 3* mit der  wird das nachfolgende Script bearbeitet.

C-Aktion

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                  char* lpszPropertyName, UINT nFlags, int x, int y)
{
//for loop
int i,j;

printf("\r\nexample 3 : for loop\r\n");

j=0;//execution count

for(i=7;i>=0;i--) {
    j++;
    printf("%d. run : i=%d\r\n",j,i);
} //for
}
```

Ausgabe im Diagnosefenster:




Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der C Variablen. Die Variable *i* wird für die Schleifenfunktion benötigt, *j* nur für die Ausgabe der Durchläufe.
- Im Diagnosefenster ausgeben, was anschließend kommt.
- Die Variable für den Schleifenzähler wird initialisiert.
- Die Bearbeitung der *for* Schleife erfolgt solange $i \geq 0$ ist. Der Startwert von *i* ist 7.
- Die Variable *j* wird beim Durchlauf incrementiert, *i* wird nach dem Durchlauf decrementiert.
- Anschließend wird der aktuelle Wert der Variablen mit der Funktion *printf* im Diagnosefenster ausgegeben.

4.5.4 Beispiel 4- Bedingte Anweisung mit if

In Beispiel 4 zeigen wir Ihnen die Verwendung der bedingten *if* Anweisung in einer *C-Aktion*.

Die Aktion ist am Button *example_04* → *Ereignis* → *Maus* → *links drücken* projiziert.

Beim Drücken des Button *Beispiel 4* mit der  wird das nachfolgende Script bearbeitet.

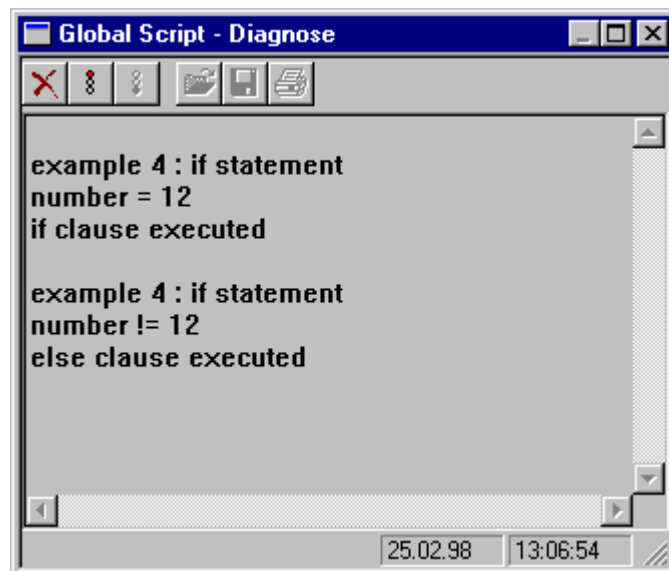
C-Aktion

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                  char* lpszPropertyName, UINT nFlags, int x, int y)
{
//if statement
static int number=12;

printf("\r\nexample 4 : if statement\r\n");

if (number==12) {
    printf("number = 12\r\n");
    printf("if clause executed\r\n");
    number=11;
} //if
else {
    printf("number != 12\r\n",number);
    printf("else clause executed\r\n");
    number=12;
} //else
}
```

Ausgabe im Diagnosefenster:




Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der C Variable als *static*. Der Wert dieser Variable bleibt bis zum nächsten Bildwechsel erhalten.
- Ausgabe im Diagnosefenster was anschließend kommt.
- Die Bearbeitung der *Abfrage*.
- Ist die Bedingung (*number==12*) erfüllt, wird der *if* Zweig bearbeitet. Bei Nichterfüllung der Bedingung wird der *else* Zweig bearbeitet.
- Das aktuelle Ergebnis der Abfrage wird mit der Funktion *printf* im Diagnosefenster ausgegeben.

4.5.5 Beispiel 5- Bedingte Anweisung mit switch und case

In Beispiel 5 zeigen wir Ihnen die Verwendung der *switch case* Anweisung in einer *C-Aktion*.

Die Aktion ist am Button *example_05* → *Ereignis* → *Maus* → *links drücken* projiziert.

Beim Drücken des Button *Beispiel 5* mit der  wird das nachfolgende Script bearbeitet.

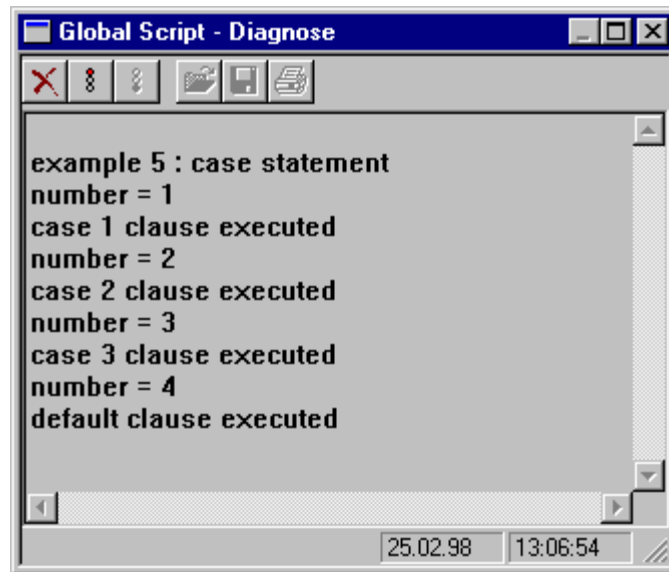
C-Aktion

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                  char* lpszPropertyName, UINT nFlags, int x, int y)
{
//case statement
static int number=1;

if (number==1) printf("\r\nexample 5 : case statement\r\n");


switch (number) {
    case 1: {
        printf("number = %d\r\n",number);
        printf("case 1 clause executed\r\n");
        number++;
        break;
    }//case1
    case 2: {
        printf("number = %d\r\n",number);
        printf("case 2 clause executed\r\n");
        number++;
        break;
    }//case2
    case 3: {
        printf("number = %d\r\n",number);
        printf("case 3 clause executed\r\n");
        number++;
        break;
    }//case3
    default : {
        printf("number = %d\r\n",number);
        printf("default clause executed\r\n");
        number=1;
        break;
    } //default
} //switch
}
```

Ausgabe im Diagnosefenster:



```
example 5 : case statement
number = 1
case 1 clause executed
number = 2
case 2 clause executed
number = 3
case 3 clause executed
number = 4
default clause executed
```

Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der C Variablen als *static*. Der Wert dieser Variable bleibt bis zum nächsten Bildwechsel erhalten.
- Beim ersten Durchlauf erfolgt im Diagnosefenster die Ausgabe, was anschließend kommt.
- Bei der *switch* Anweisung wird die zu untersuchende Variable festgelegt (*number*).
- Die *case* Anweisungen überprüfen, ob die Bedingung erfüllt ist. Die Variable wird auf Gleichheit überprüft. Bei Nichterfüllung der Bedingung wird die nächste *case* Bedingung überprüft. Ist dieser Zweig ebenfalls nicht erfüllt, wird die nächste *case* Bedingung überprüft usw.
- Bei jedem Drücken des Buttons mit der  wird eine *case* Anweisung oder die *default* Anweisung bearbeitet.
- Mit *printf* erfolgt die Ausgabe im Diagnosefenster.
- Der Wert der Variable *number* wird verändert.

4.5.6 Beispiel 6- Verwendung statischer Variablen mit bedingter Anweisung und Rückgabewert

In Beispiel 6 zeigen wir Ihnen die Verwendung von C Variablen vom Typ *static* in Verbindung mit einer bedingten Anweisung. Die Veränderung der Schriftgröße erfolgt über den Rückgabewert in einer *C-Aktion*.

Die Aktion ist an der Eigenschaft *static_text_01* → *Eigenschaften* → *Schrift* → *Zeichensatzgröße* projiziert. Das nachfolgende Script wird bei angewähltem Bild im Zyklus von 250ms bearbeitet.

C-Aktion

```
#include "apdefap.h"
long _main(char* lpszPictureName, char* lpszObjectName, char*
           lpszPropertyName)
{
    Static int i=0;
    Static BOOL a=0;

    If ((i<1)|| (i>60)) a=!a;

    //inc or dec i according to a
    if (a) i=i+6;
    else i-=6;//short form for i=i-6

    //return
    return(i+20);
}
```



Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der C Variablen als *static*. Der Wert dieser Variablen bleibt bis zum nächsten Bildwechsel erhalten. Beide werden mit 0 initialisiert.
- Der Wert der Variablen *a* wird bestimmt. Immer wenn *i* den zulässigen Bereich verläßt, wird der Wert der Variablen *a* invertiert.
- Abhängig von der Variablen *a* wird die Variable *i* incrementiert oder decrementiert. Ist die Bedingung erfüllt, wird incrementiert Eine andere Schreibweise ist *if (a!=0)*.
- Zur Berechnung der Schriftgröße wird die Mindestgröße von 20 Punkten zur Variablen *i* addiert und mit *return* als Rückgabewert an die Eigenschaft *Schriftgröße* des Objekts übergeben.

4.5.7 Beispiel 7- Verwendung statischer Variablen mit bedingter Anweisung und Rückgabewert

In Beispiel 7 zeigen wir Ihnen die Verwendung von C Variablen vom Typ *static* in Verbindung mit einer bedingten Anweisung. Die Veränderung der Schriftfarbe erfolgt über den Rückgabewert in einer *C-Aktion*.

Die Aktion ist an der Eigenschaft *static_text_02* → *Eigenschaft* → *Farben* → *Schriftfarbe* projiziert. Das nachfolgende Script wird bei angewähltem Bild im Zyklus von 250ms bearbeitet.

C-Aktion

```
#include "apdefap.h"
long _main(char* lpszPictureName, char* lpszObjectName, char*
           lpszPropertyName)
{
static long int i=0;
static BOOL    a=0;

if ((i<400)|| (i>10000000)) a=!a;

//inc or dec i according to a
if (a==1) i+=500; //short form for i=i+500
else i=i-500;

//return color
return (i+100000);
}
```

3

C-Script über rechte Maus

Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der C Variablen als *static*. Der Wert dieser Variablen bleibt bis zum nächsten Bildwechsel erhalten.
- Immer wenn *i* den zulässigen Bereich verläßt, wird der Wert der Variablen *a* invertiert.
- Abhängig von der Variable *a* wird die Variable *i* incrementiert oder decrementiert.
- Für die Rückgabe des Farbwertes wird der Wert 100000 zur Variablen *i* addiert und an die Eigenschaft des Objekts übergeben. Hinweise zur Farbkodierung finden Sie im Anhang im Kapitel 5.1.6 *Farbtabelle*.

4.6 Dateioperationen in C


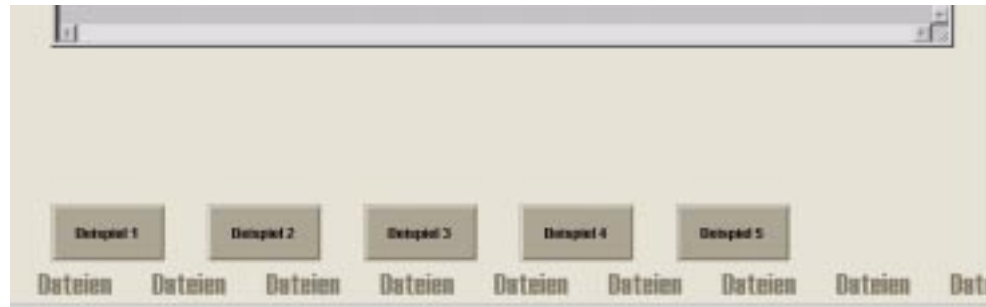


In unserem Projekt erreichen Sie die Beispiele zum Thema Dateien durch die Anwahl des Buttons *Dateien*. Mit der  wird der Sourcecode zum jeweiligen Beispiel angezeigt.

Bild für Dateien



Nach der Anwahl des Button *Dateien* mit der  sehen Sie das oben dargestellte Bild. Die einzelnen Button im Bild führen die beschriebenen Beispiele aus. Mit der  wird der Sourcecode zum Beispiel angezeigt.

Bevor wir mit den Beispielen beginnen, etwas Theorie zum Thema Dateioperationen.
In C ist eine Datei eine Ansammlung von Zeichen, unabhängig vom Inhalt.
Bevor man in C eine Datei bearbeiten (lesen, schreiben) kann, muß man sie *öffnen*.

fopen

Mit der Funktion *fopen* wird eine Datei geöffnet. Als Argument wird die zu bearbeitende Datei und der Bearbeitungsmodus übergeben.

```
FILE *datei;
datei = fopen (Dateiname , Arbeitsmodus);
```

Der Dateiname kann auch mit Pfadangaben übergeben werden.
Tritt ein Fehler auf, so wird keine Verbindung zu einer Dateibeschreibung, d.h. in C ein *NULL-Pointer*, zurückgeliefert. In der Variablen des Typs *FILE* stehen die Informationen, die Dateibefehle zur Bearbeitung der Datei benötigen.

Im Bearbeitungsmodus bestimmen wir, welche Funktionen erlaubt sind.

Modus	Beschreibung
r	Öffnen der Datei zum Lesen (r = read) Rückgabe ist NULL, wenn die Datei nicht existiert, oder keine Leserechte bestehen.
w	Öffnen der Datei zum Schreiben (w = write) Rückgabe ist NULL, wenn die Datei READ-ONLY ist, oder nicht existiert.
a	Öffnen der Datei zum Anhängen an das Ende (a = append) Wenn die Datei existiert, wird das Geschriebene an das Ende gehängt. Wenn die Datei nicht existiert, wird sie angelegt. Rückgabe ist NULL, wenn keine Datei erstellt werden darf oder die Datei nicht beschrieben werden darf.
r+	Die Datei wird zum abwechselnden Lesen und Schreiben geöffnet. Rückgabe ist NULL, wenn die Datei nicht existiert, oder keine Lese- und Schreibrechte für die Datei existieren.
w+	Anlegen einer neuen Datei zum abwechselnden Lesen und Schreiben. Existiert die Datei schon, so wird sie gelöscht! Rückgabewert ist NULL, wenn der Ersteller keine Rechte für diese Aktionen hat.
a+	Öffnen einer Datei zum Lesen oder zum Schreiben (anhängen) am Ende. Die Datei wird erstellt, falls sie nicht existiert. Rückgabe ist NULL, wenn eine Datei nicht gelesen oder beschrieben werden darf, bzw. wenn die Dateirechte nicht ausreichend sind.

fclose

Mit *fclose* werden Dateien, die wir zuvor mit *fopen* geöffnet wurden wieder geschlossen.

```
fclose (datei);
```

fscanf

Der Befehl verhält sich genau wie *scanf*, mit dem Unterschied, daß hier angegeben wird, aus welcher Datei die Information kommt.

```
fscanf ( datei , "%c" , zeichen);
```


fprintf

Der Befehl verhält sich genau wie *printf*, mit dem Unterschied, daß hier angegeben wird, in welche Datei die Information kommt.

4.6.1 Beispiel 1 - Datei öffnen, beschreiben und schließen

In Beispiel 1 zeigen wir die grundlegenden Dateioperationen in C.

Die Aktion ist am Button *example_01* → Ereignis → Maus → links drücken projiziert.

Beim Drücken des Button *Beispiel 1* mit der  wird das nachfolgende Script bearbeitet.

C-Aktion

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
    FILE *datei;
    long l;
    float fp;
    char s[20];
    char c;

    //open file to write
    datei = fopen( "cours_00.dat", "w+" );

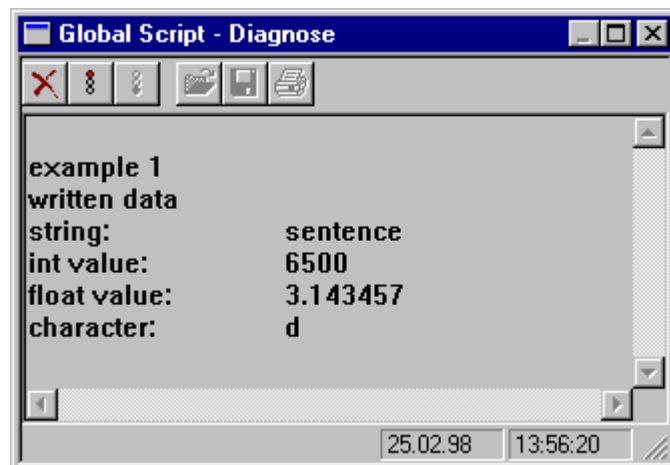
    //file open error ??
    if( datei != NULL ){
        l=6500;
        strcpy(s,"sentence");
        fp=3.1434567;
        c='d';

        //write data
        fprintf( datei,"%s\r\n%d\r\n%f\r\n%c\r\n", s, l, fp, c );

        //output in diagnostics window
        printf("\r\nexample 1\r\n");
        printf("written data\r\n");
        printf("string:\t\t%s\r\n",s);
        printf("int value:\t\t%d\r\n",l);
        printf("float value:\t\t%f\r\n",fp);
        printf("character:\t\t%c\r\n",c);

        //close file
        fclose( datei );
    }//if
    else printf( "file open error\r\n" );
}
```

Ausgabe im Diagnosefenster:




Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der Variablen und des Zeigers.
- Das Öffnen der Datei zum Überschreiben. Ist die Datei noch nicht vorhanden wird sie angelegt.
- In diesem Abschnitt wird in einer bedingten Abfrage das Öffnen der Datei überprüft. Wenn dies funktioniert, werden den Variablen die Werte zugewiesen und mit *fprintf* in die Datei geschrieben. Wenn das Öffnen oder Anlegen der Datei nicht funktioniert hat, wird der *else* Zweig bearbeitet.
- Anschließend werden die Werte mit der Funktion *printf* im Diagnosefenster ausgegeben.
- Im letzten Schritt wird die Datei geschlossen.

4.6.2 Beispiel 2 - Datei öffnen, erweitern und schließen

In Beispiel 2 zeigen wir Ihnen die grundlegenden Dateioperationen in C.

Die Aktion ist am Button *example_02* → Ereignis → Maus → links drücken projiziert.

Beim Drücken des Button *Beispiel 2* mit der  wird das nachfolgende Script bearbeitet.

C-Aktion

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                  char* lpszPropertyName, UINT nFlags, int x, int y)
{
    static int    i = 0;
    FILE         *datei;
    Char         s[200];

    switch (i) {
        case 0 : strcpy(s, "This\r\n");    break;
        case 1 : strcpy(s, "sentence\r\n"); break;
        case 2 : strcpy(s, "is\r\n");      break;
        case 3 : strcpy(s, "a\r\n");       break;
        case 4 : strcpy(s, "long\r\n");    break;
        case 5 : strcpy(s, "one\r\n");     break;
    }

    //open file to write and append data
    datei = fopen("cours_00.dat", "a");

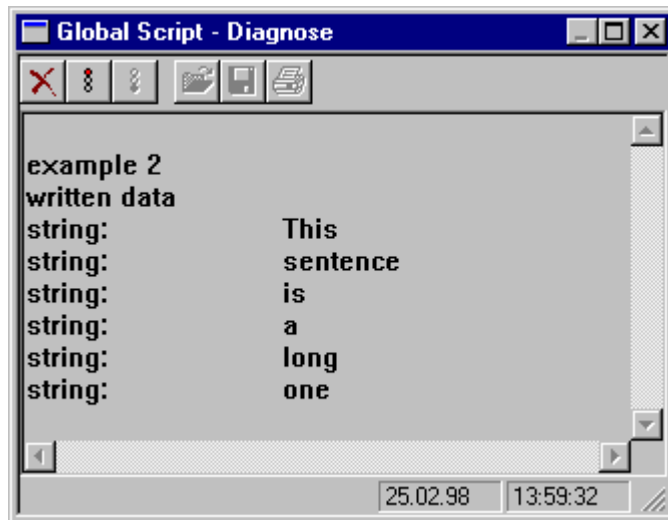
    //file open error ??
    if (datei != NULL){

        //write data
        fprintf(datei, "%s", s);

        //output in diagnostics window
        if (i==0) {
            printf("\r\nexample 2");
            printf("\r\nwritten data\r\n");
        }
        printf("string:\t\t%s", s);

        fclose( datei );//close file
    }//if
    else printf("\r\nfile open error\r\n");
    i++;
    if (i>5) (i=0);
}
}
```

Ausgabe im Diagnosefenster:



```
Global Script - Diagnose
example 2
written data
string:      This
string:      sentence
string:      is
string:      a
string:      long
string:      one
25.02.98    13:59:32
```


Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der Variablen und des Zeigers.
- Generieren eines Textes in Abhängigkeit der statischen Variable *i*.
- Das Öffnen der Datei zum Anhängen der Daten an das Dateieinde.
- In diesem Abschnitt wird in einer bedingten Abfrage das Öffnen der Datei überprüft. Wenn das Öffnen funktioniert hat, wird mit *fprintf* der aktuelle Text in die Datei geschrieben. Die Daten (Inhalt der Variable *s*) werden an das Dateieinde angehängt.
- Anschließend wird der geschriebene Text mit der Funktion *printf* im Diagnosefenster ausgegeben.
- Im letzten Schritt wird die Datei geschlossen.

4.6.3 Beispiel 3 - Datei öffnen, lesen und schließen

In Beispiel 3 zeigen wir Ihnen die grundlegenden Dateioperationen in C.

Die Aktion ist am Button *example_03* → *Ereignis* → *Maus* → *links drücken* projiziert.

Beim Drücken des Button *Beispiel 3* mit der  wird das nachfolgende Script bearbeitet.

C-Aktion

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                  char* lpszPropertyName, UINT nFlags, int x, int y)
{
    FILE *datei;
    int i;
    char t[20];
    char* z;
    //open file to read
    datei = fopen("cours_00.dat","r");

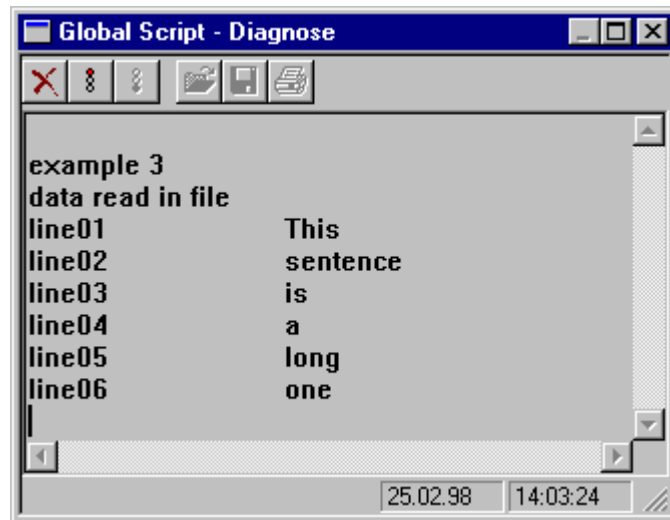
    //file open error ??
    if (datei !=NULL) {

        printf("\r\nexample 3");
        printf("\r\n\data read in file\r\n");

        //read data in file
        i=1;
        while (i) {
            i++;
            z=fgets(t,20,datei);
            if (z==NULL) break;//end of file
            //output in diagnostics window
            printf("line%02d\t\t%s", (i-1),t);
        }//while

        //close file
        fclose( datei);
    }//if
    else{
        printf("\r\nfile open error\r\n" );
    }//else
}
```

Ausgabe im Diagnosefenster:



```
example 3
data read in file
line01      This
line02      sentence
line03      is
line04      a
line05      long
line06      one
```

25.02.98 14:03:24


Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der Variablen und Zeiger.
- Das Öffnen der Datei.
- In diesem Abschnitt wird in einer bedingten Abfrage das Öffnen der Datei überprüft. Wenn dies funktioniert hat, werden die Daten mit *fgets* zeilenweise aus der Datei gelesen und mit der Funktion *printf* im Diagnosefenster ausgegeben. Bei Erreichen vom Dateiende (Rückgabewert ist 0) wird die *while* Schleife mit *break* verlassen.
- Zum Schluß wird die Datei geschlossen.

4.6.4 Beispiel 4 - Datei löschen

In Beispiel 4 zeigen wir Ihnen das Löschen der zuvor angelegten Datei.

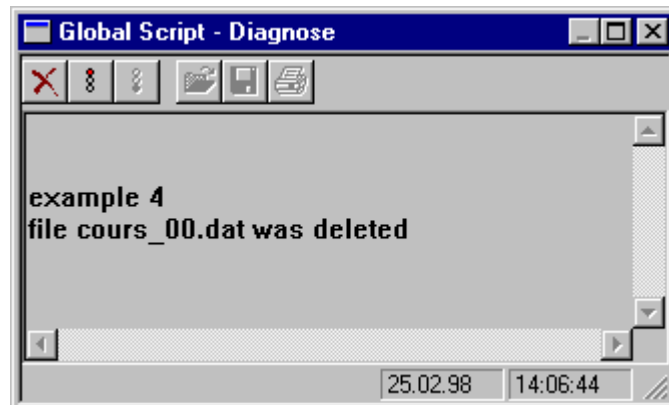
Die Aktion ist am Button *example_05* → *Ereignis* → *Maus* → *links drücken* projiziert.

Beim Drücken des Button *Beispiel 5* mit der  wird das nachfolgende Script bearbeitet.

C-Aktion

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
    //delete file
    remove("cours_00.dat");
    printf("\r\n\r\nexample 4\r\n");
    printf("file cours_00.dat was deleted\r\n");
}
```

Ausgabe im Diagnosefenster:



Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Datei wird gelöscht.
- Im Diagnosefenster wird ausgegeben was passiert ist.

4.7 Strukturen in C


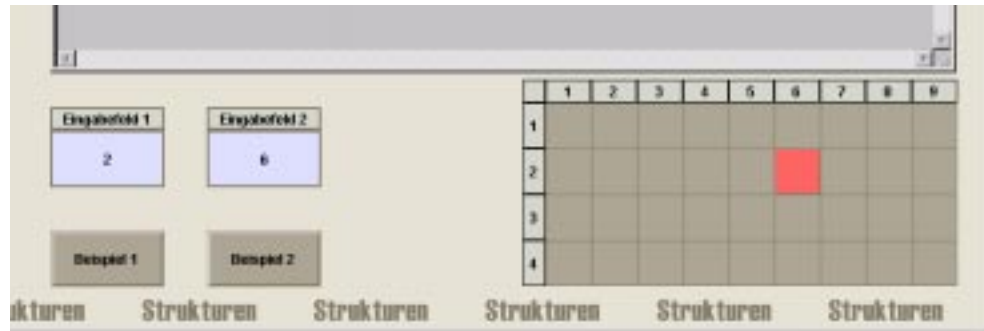


In unserem Projekt erreichen Sie die Beispiele zum Thema durch die Anwahl des Button *Strukturen*. Mit der  wird der Sourcecode zum jeweiligen Beispiel angezeigt.

Bild für Strukturen



Nach der Anwahl des Button *Strukturen* mit der  sehen Sie das oben dargestellte Bild. Die einzelnen Button im Bild führen die beschriebenen Beispiele aus. Mit der  wird der Sourcecode zum Beispiel angezeigt. Im rechten Feld werden die Werte der *Eingabefelder* im *Beispiel 2* ausgewertet.

Bevor wir mit den Beispielen beginnen, etwas Theorie zum Thema Strukturen.

Definition der Struktur

Prinzipiell sieht eine Strukturdeklarierung in C wie folgt aus.

```
struct Strukturname {
    [ Komponenten ]
};
```

Komponenten sind hier eine Ansammlung von definierten Variablen, die selbst Strukturelemente sind und fest zu der erstellten Struktur gehören.

Mit der Struktur können wir im Prinzip wie mit einem Variablentypen umgehen, da es sich hierbei um eine Ansammlung von Datentypen handelt.

Mit

```
struct ganz {
    int zahl;
    char zeichen[50];
};

struct ganz struktur[2];
```

legen wir im Vereinbarungsteil den Strukturaufbau fest.

Man kann sich vorstellen, das ein Datentyp *struct ganz* erzeugt wird und die Variable *struktur* vom Typ *struct ganz* ist.

Bei der Deklaration gibt es folgende Vereinfachung

Die Deklaration des Strukturaufbaus und die Definition einer Variable kann zu einer Anweisung zusammengefaßt werden.

```
struct ganz {
    int zahl;
    char zeichen[50];
} struktur[2];
```

Zugriff auf die Variablen in der Struktur

Mit *Strukturname.Variablenname* kann man gezielt auf einzelne Daten zugreifen. Es ist wichtig, daß zwischen den Elementen der Punkt steht. Er trennt in der Hierarchie die einzelnen Elemente.

Der Variablenname ist immer der Name, der in der Strukturerstellung benutzt wird!
Mit

```
struktur[0].zahl = 2;
strcpy(struktur[0].zeichen, "unsere erste Struktur");
```

weisen wir den Elementen die Werte zu. Dabei ist stets darauf zu achten, daß der Datentyp korrekt ist (Mit *strcpy* wird einer Zeichenkette einer Zeichenfolge zugewiesen). Man kann wie bisher auch, auf Felder innerhalb von Strukturen zugreifen:

```
Printf ("%c", struktur[0].zeichen[5] );
```

gibt uns z.B. den 6. Buchstaben der Zeichenkette aus dem 1. Feld aus *struktur [0]* aus.

Indizes fangen bei 0 an !

Zeiger als Strukturelemente

Innerhalb einer Struktur können beliebige Variablentypen stehen. Das schließt auch Zeiger mit ein.

Folgende Struktur ist also durchaus denkbar:

```
struct struktur {
    int    zahl;
    char   *zeichenkette;
}
```

Hinweis:

Wenn ein Zeiger auf eine Zeichenkette (z.B. *zeichenkette) verwendet wird, muß der Speicher auch verfügbar sein oder vorher allokiert (z.B. mit der internen Funktion *sysMalloc (8)*) werden.

Als Beispiel benutzen wir die Struktur des vorigen Beispiels und generieren folgenden Zeiger:

```
struct struktur *zeiger;
```

Wenn wir nun auf ein Element zugreifen wollen, müssen wir

```
(*zeiger).zahl; // oder Einfacher zeiger->zahl;
```

schreiben.


Hinweis:

Da der Punkt eine größere Priorität gegenüber dem * Operator hat, muß der erste Ausdruck in Klammern gesetzt werden.

4.7.1 Beispiel 1 - Strukturen in C

In Beispiel 1 zeigen wir Ihnen die Verwendung von Strukturen in C.

Die Aktion ist am Button *example_01* → Ereignis → Maus → links drücken projiziert.

Beim Drücken des Button *Beispiel 1* mit der  wird das nachfolgende Script bearbeitet.

C-Aktion

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
    //declare structure
    struct daten {
        char name[40];
        int age;
        int weight;
        float height; } family[3];

    int    i;

    //set values to structure elements
    strcpy(family[0].name, "Rudi");
    strcpy(family[1].name, "Simone");
    strcpy(family[2].name, "Bernhard");

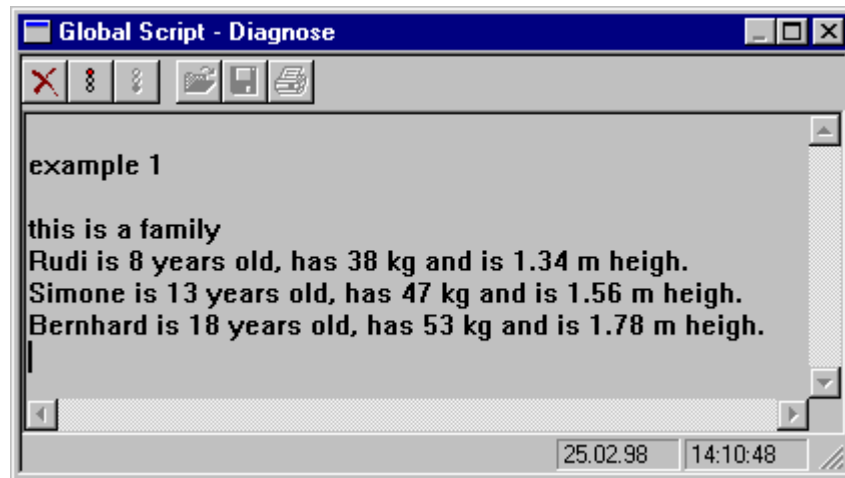
    family[0].age=8;
    family[1].age=13;
    family[2].age=18;

    family[0].weight=38;
    family[1].weight=47;
    family[2].weight=53;

    family[0].height=1.34;
    family[1].height=1.56;
    family[2].height=1.78;

    //output in diagnostics window
    printf("\r\nexample 1\r\n");
    printf("\r\nthis is a family\r\n");
    for (i=0;i<3;i++) {
        printf("%s is %d years old, has %d kg and is %.2f m
        heigh.\r\n",family[i].name,family[i].age,family[i].weight,family[i].
        height);
    }
}
```



Ausgabe im Diagnosefenster:



Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der Struktur und der Variable.
- Den Strukturelementen die Werte zuweisen.
- Anschließend werden der Werte (Informationen) mit der Funktion *printf* im Diagnosefenster ausgegeben.

4.7.2 Beispiel 2 - C Strukturen in Verbindung mit WinCC

In Beispiel 2 zeigen wir Ihnen die Verwendung von Strukturvariablen in WinCC. Bevor wir diese Strukturvariablen in unserer Aktion verwenden können, müssen diese im Variablenhaushalt des *Control Centers* angelegt werden. Eine ausführliche Erklärung finden Sie im *ConfigurationManual Band 2* im *Beispiel 2.8 Strukturvariablen verwenden*. Die Aktion ist am Button *example_02* → *Ereignis* → *Maus* → *links drücken* projiziert. Beim Drücken des Button *Beispiel 2* mit der  wird das nachfolgende Script bearbeitet.

C-Aktion

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                  char* lpszPropertyName, UINT nFlags, int x, int y)
{
int input1,input2;
int output1,output2;

//read selected position in input_01 and input_02
input1=GetTagDWord("S32i_course_str_01");
input2=GetTagDWord("S32i_course_str_02");

input1=input1-1;
input2=input2-1;

//calculate new rectangle position
output1=500+(input1*40);
output2=470+(input2*40);

//set internal tags which contain rectangle position
SetTagDWord("STUi_course_str_00.xpos",output2);
SetTagDWord("STUi_course_str_00.ypos",output1);
}
```

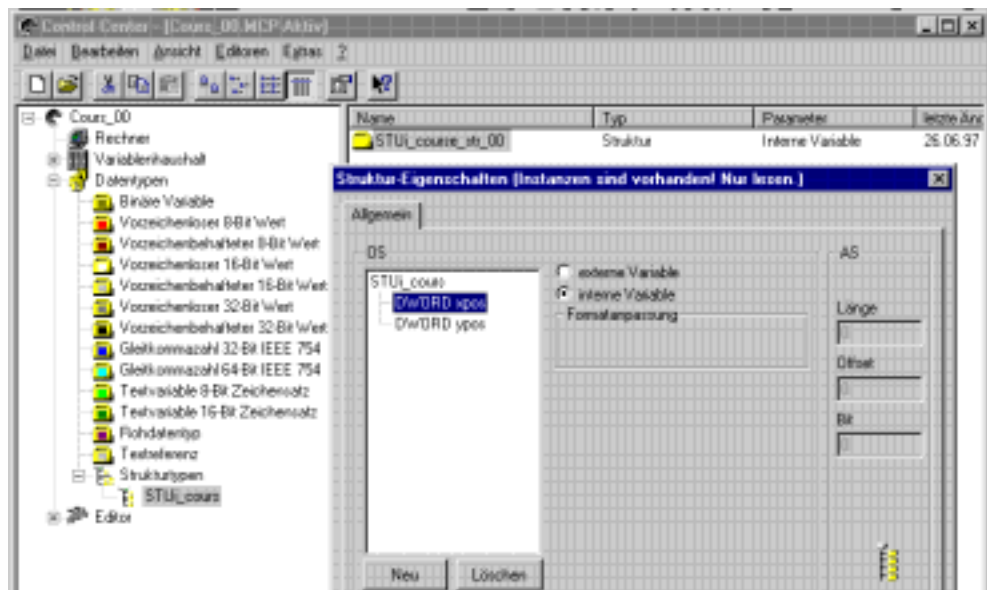
Anzeige am Bildschirm:

	1	2	3	4	5	6	7	8	9
1									
2									
3									
4									

Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der Variablen.
- Auslesen der Werte aus den EA-Feldern. Die Grenzen der Eingabe sind direkt bei den *EA-Feldern* festgelegt.
- Die eingegebenen Positionsangaben in Bildschirmpositionen umrechnen.
- Die errechneten Werte der WinCC Strukturvariablen zuweisen. Die Werte sind mit der X und Y-Position des Rechtecks verbunden.
- Die verwendete Struktur muß im *Control Center* → *Datentypen* → *Strukturtypen* deklariert werden und hat folgenden Aufbau:

Anlegen der Strukturvariablen im Variablenhaushalt



4.8 Global Scripts


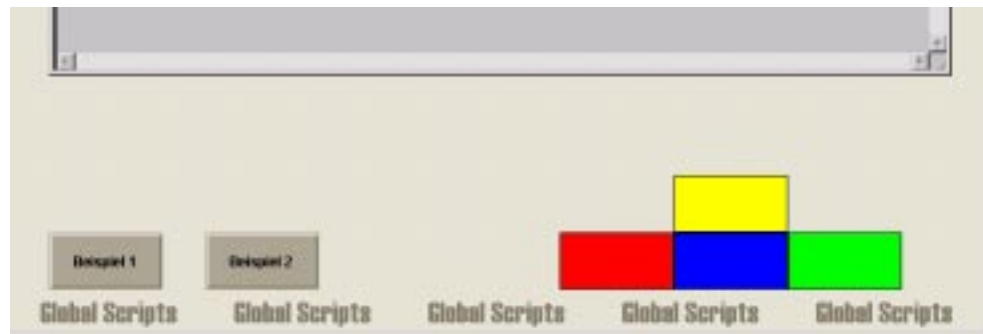


In unserem Projekt erreichen Sie die Beispiele zum Thema *Global Scripts* durch die Auswahl des Buttons *GSC*. Mit der  **R** wird der Sourcecode zum jeweiligen Beispiel angezeigt.


Bild für Global Script



Nach der Auswahl des Button *GSC* mit der  sehen Sie das oben dargestellte Bild. Die einzelnen Button im Bild führen die beschriebenen Beispiele aus. Mit der  **R** wird der Sourcecode zum Beispiel angezeigt.

Die Projektfunktionen wurden im Editor *Global Script* erstellt und in den objektgebundenen Aktionen im *Graphics Designer* eingesetzt. Die Auswahl der Funktionen erfolgt über den Funktionsbaum im Editor der *C-Aktion*.

4.8.1 Beispiel 1 - Verwenden einer Projektfunktion

In Beispiel 1 zeigen wir Ihnen die Verwendung von Projektfunktionen in einer C-Aktion. Die Aktion ist am Button *example_01* → *Ereignis* → *Maus* → *links drücken* projiziert. Beim Drücken des Button *Beispiel 1* mit der  wird das nachfolgende Script bearbeitet.

Programm der Projektfunktion

```
void calculate(float a,float b,float *pRes1,float *pRes2)
{
//calculate and store value in addresses
*pRes1=a/b;    //divide
*pRes2=a*b;    //multiply
}
```

Programm der Aktion am Objekt

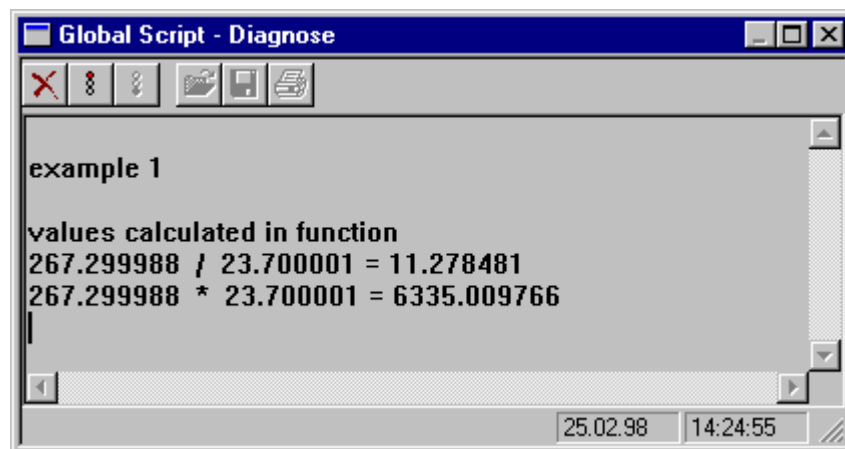
```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
char* lpszPropertyName, UINT nFlags, int x, int y)
{
float number1,number2;
float result1,result2;

number1=267.3;
number2=23.7;

//execute function and set parameters
calculate(number1,number2,&result1,&result2);

//output in diagnostics window
printf("\r\nexample 1\r\n");
printf("\r\nvalues calculated in function\r\n");
printf("%f / %f = %f\r\n",number1,number2,result1);
printf("%f * %f = %f\r\n",number1,number2,result2);
}
```


Ausgabe im Diagnosefenster:



Erklärung der einzelnen Programmteile

- Im Programm der Aktion werden die Variablen (*number1*, *number2*) deklariert und der Projektfunktion *calculate* übergeben.
- Im Programm der aufgerufenen Projektfunktion, werden die übergebenen Werte berechnet. Die Rückgabe der berechneten Werte erfolgen über die beiden Zeiger **pRes1*, **pRes2*
- Anschließend werden die Werte in der *C-Aktion* mit der Funktion *printf* im Diagnosefenster ausgegeben.

4.8.2 Beispiel 2 - Verwenden von Projektfunktionen, weitere Beispiele

In Beispiel 2 zeigen wir Ihnen die Verwendung von Projektfunktionen in einer C-Aktion. Die Aktion ist am Button *example_02* → *Ereignis* → *Maus* → *links drücken* projektiert. Beim Drücken des Button *Beispiel 2* mit der  wird das nachfolgende Script bearbeitet.

Programm der Projektfunktion

```
void colourchange()
{
long int color1,color2,color3,color4;
long int store;

//get rectangle colours
color1=GetBackColor(lpszPictureName,"rectangle_01");
color2=GetBackColor(lpszPictureName,"rectangle_02");
color3=GetBackColor(lpszPictureName,"rectangle_03");
color4=GetBackColor(lpszPictureName,"rectangle_04");

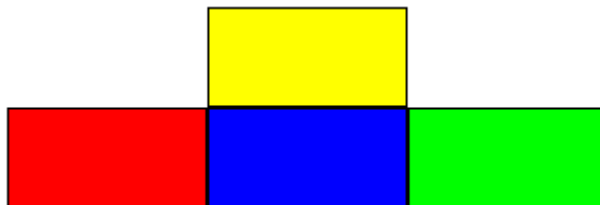
//cyclic colour change
store=color1;
color1=color2;
color2=color3;
color3=color4;
color4=store;

//set rectangle colours
SetBackColor(lpszPictureName,"rectangle_01",color1);
SetBackColor(lpszPictureName,"rectangle_02",color2);
SetBackColor(lpszPictureName,"rectangle_03",color3);
SetBackColor(lpszPictureName,"rectangle_04",color4);
}
```

Programm der Aktion am Objekt

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
char* lpszPropertyName, UINT nFlags, int x, int y)
{
//execute function
colourchange(lpszPictureName);
}
```

Änderung der Farben bei Mausklick:



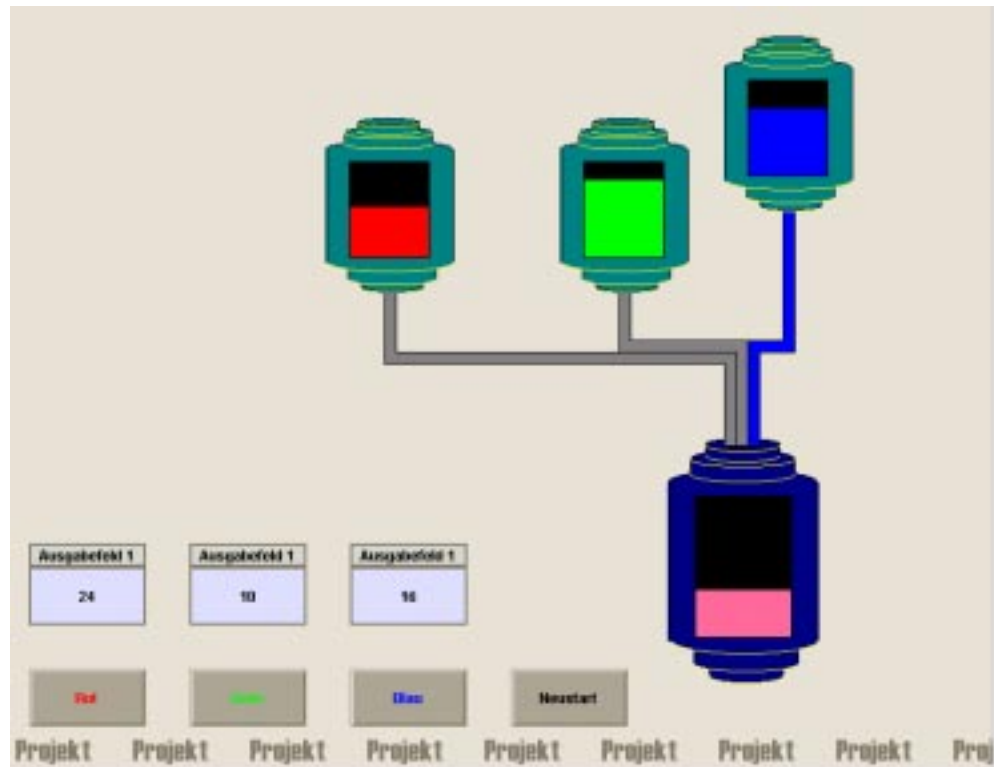
Erklärung der einzelnen Programmteile


- Im Programm der Projektfunktion werden die aktuellen Farben ausgelesen, anschließend um eine Position weiter verschoben und den Objekten wieder zugewiesen.
- Beim Aufruf der Projektfunktion wird der Bildname übergeben.

4.9 Projekt als Beispiel


Dieses Projekt ist ein weiteres Beispiel für den Einsatz der zuvor beschriebenen Themen. Die einzelnen Programmteile sind hier nicht näher beschrieben.

Bild für Projekt



Nach der Auswahl des Bildes mit dem Projekt sehen Sie das oben dargestellte Bild. Dies stellt eine Farbmischanlage dar. Durch *Drücken* der Buttons *Rot*, *Grün* und *Blau* mit der  wird die entsprechende Farbe zudosiert. Erneutes *Drücken* beendet die Zudosierung. Mittels *Neustart* wird die Anlage in den Anfangszustand zurückgesetzt. In den Ausgabefeldern wird die bereits abgefüllte Menge (maximal 50) angezeigt.

4.10 Einblenden des Sourcecode mit der rechten Maus

In Projekt *Cours_00* können bei allen Beispielen die Sourcen der *C-Aktion* direkt im Runtime eingeblendet werden. Die Anzeige erfolgt, bei allen Beispielen, durch Drücken des jeweiligen Button mit der .

```
#include "apdefap.h"
void OnRButtonDown(char* lpszPictureName, char* lpszObjectName,
                  char* lpszPropertyName, UINT nFlags, int x, int y)
{
char name[30], namex[30];
int visx, comp;
int number;
int ch = '_';
char *pdest;

visx=GetVisible("course_0_startpicture_00.PDL", "code picture");
strcpy(namex, GetTagChar("T16x_org_picname_02"));

pdest = strrchr( lpszObjectName, ch );
if ( pdest == NULL )(printf("ObjectNameError"));
else {
    number = atoi(strrchr(lpszObjectName, '_')+1);
    sprintf(name, "course_4_tag_%02d.PDL", number);
    SetTagChar("T16x_org_picname_02", name);
}

comp=strcmp(name, namex);
if ((comp==0)&&(visx==1)){
    SetVisible("course_0_startpicture_00.PDL", "code picture", 0);
    SetVisible("course_0_startpicture_00.PDL", "code picture", 1);
}
else SetVisible("course_0_startpicture_00.PDL", "code picture", 1);
}
```

Erklärung der einzelnen Programmteile

- Der erste Abschnitt ist der *Funktionskopf*. Er kann nicht verändert werden.
- Die Deklaration der Variablen.
- Überprüfen ob das Bildfenster bereits angewählt ist (sichtbar).
- Den Bildnamen auslesen.
- Die Buttonnummer bestimmen.
- Den Namen des Sourcecodebildes zusammenstellen.
- Vergleich auf Änderung des Bildnamen.
- Bei gleichem Namen das Bild schließen und neu aufbauen damit es in den Vordergrund kommt.
- Bei einem neuen Bild wird dieses angezeigt.

5 Anhang

Im Anhang finden Sie eine Sammlung der Themen die nicht direkt im *Configuration Manual* eingearbeitet wurden.

5.1 Tips und Tricks

Weitere Beispiele zur Projektierung mit WinCC.

5.1.1 Normierte Ein-/Ausgabe am EA-Feld

Damit ein EA-Feld den Wert normiert anzeigt, bzw. der eingegebene Wert normiert an die Steuerung übergeben wird, sind folgende Aktionen zu projektieren:

Aktion an Property "Ausgabewert" eines EA-Feldes (wichtig: "float", wenn Nachkommastellen erwünscht)

```
Float a;  
a=GetTagFloat("DB21_DW1");  
return(a/100);
```

Aktion an Ereignis "Eingabewert" eines EA-Feldes (Variable "Var1" ist ein Vorzeichenloser 16-Bit-Wert)

```
float a;  
a=GetInputValueDouble(lpszPictureName, lpszObjectName);  
SetTagFloat("Var1", a*100);
```

5.1.2 Objektspezifische Aktionen bei Bildanwahl

Es gibt Anwendungsfälle, in denen Aktionen am Property eines oder mehrerer Objekte in einem Bild nur einmalig bei Bildanwahl durchzuführen sind. Eine Möglichkeit ist, eine bildspezifische Aktion am Bildobjekt unter *Ereignisse* → *Sonstige* → *Bildanwahl* zu formulieren. Dies hat aber Nachteile, daß die Aktion auf Objekte im Bild wirken muß und dadurch in der Aktion die Objektnamen fix genannt werden müssen. Die Objekte sind nicht mehr frei hantierbar. Diese Lösung ist nicht objektorientiert.

Es gibt eine Möglichkeiten, dieses Problem zu umgehen:

- Definieren Sie eine interne Variable (z.B. *dummy*), die nie aktualisiert oder gezielt gesetzt wird. Triggern Sie die Aktion am Objekt auf Änderung dieser Variable. Beim Aufschlagen des Bildes im Runtime wird die Aktion erstmalig aktiviert und würde danach nur noch bei Änderung der Variable *dummy* erneut reagieren, was aber nicht erfolgt, da diese Variable nie geändert wird.

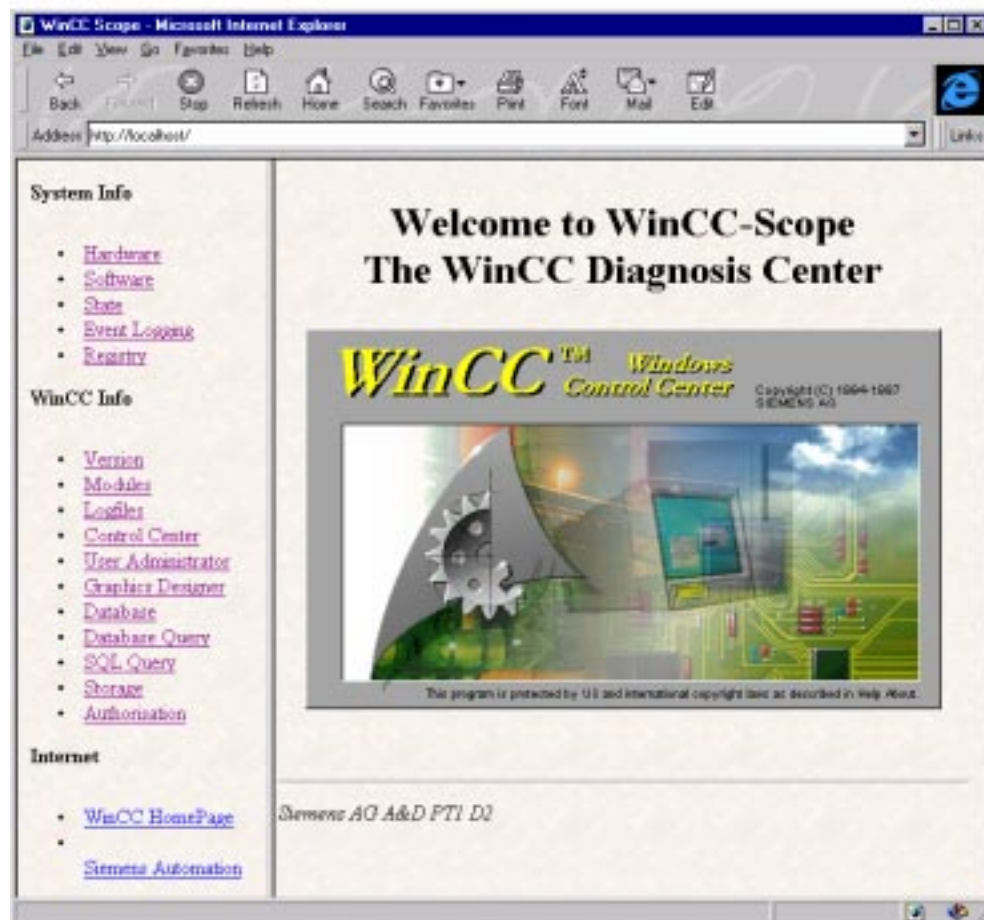
5.1.3 WinCC-Scope

Allgemeines

WinCC-Scope ist ein Werkzeug, welches Sie bei der Diagnose von WinCC-Projekten unterstützt. Es stellt Ihnen eine Vielzahl an Informationen über das aktivierte Projekt, aber auch über das jeweilige Computersystem zur Verfügung. Um mit *Scope* arbeiten zu können, wird ein Webbrowser wie der Internetexplorer benötigt. Weiters muß als Netzwerkprotokoll TCP/IP installiert sein.

Start und Bedienung

Haben Sie WinCC installiert, wurde auch standardmäßig *Scope* installiert. Bevor *Scope* benutzt werden kann, muß das Programm *WinCCDiagAgent.exe* gestartet werden. Es befindet sich im Verzeichniss *Siemens\WinCC\WinCCScope\bin*. Es handelt sich hierbei um einen einfachen HTTP-Server. Danach kann *Scope* über das Startmenü aktiviert werden. Auf der Startseite wird über den Verweis *How to use the new Diagnostics Interface* eine Seite mit einer allgemeinen Beschreibung über die Bedienung von *WinCC-Scope* erreicht. Klicken Sie auf den Verweiß *http://localhost*, um *Scope* zu starten. Über die Liste im linken Fenster können Informationen verschiedenster Art abgerufen werden. Im Abschnitt *System Info* werden allgemeine Informationen zum jeweiligen Computersystem erreicht, im Abschnitt *WinCC Info* werden Informationen zum gerade aktivierten WinCC-Projekt erreicht.

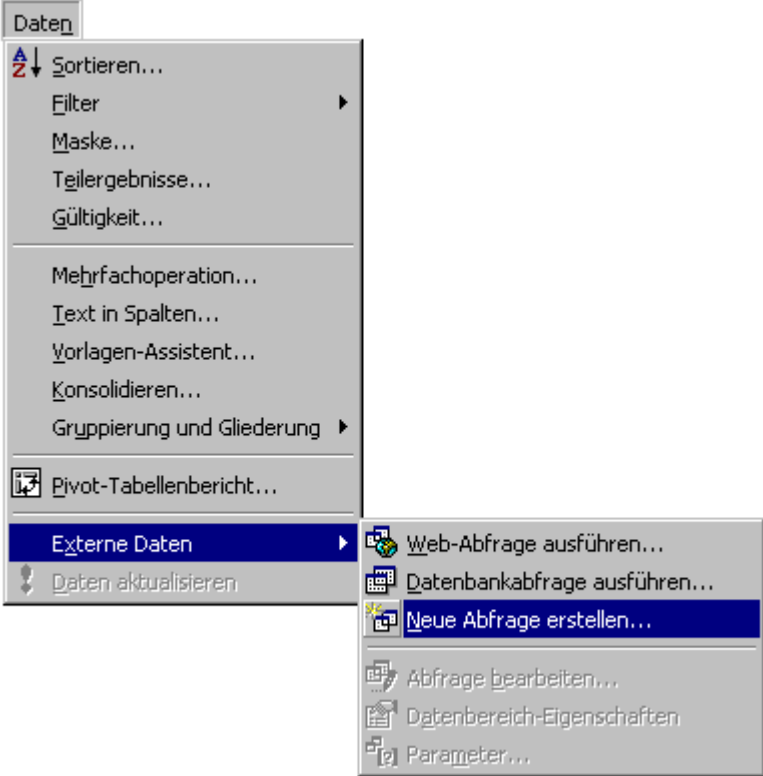


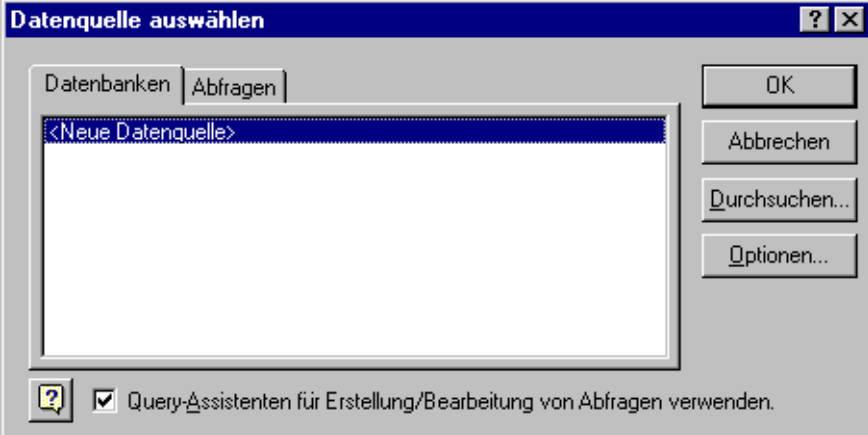
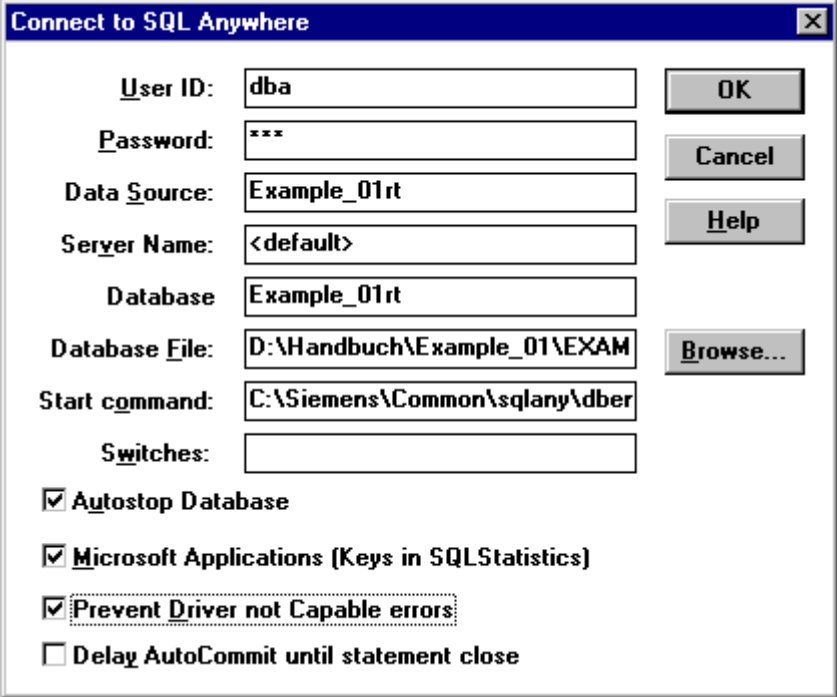
5.1.4 Datenbank-Zugriffe

5.1.4.1 Zugriff auf die Datenbank von Excel/MSQuery aus


Die folgende Beschreibung für den Zugriff auf die WinCC-Datenbank bezieht sich auf die Verwendung von Microsoft® Excel 97 mit SR-1.

Zugriff von Excel/MSQuery

Schritt	Vorgehen: Zugriff von Excel/MSQuery
1	<p>Excel öffnen. Über den Menüpunkt <i>Daten</i> → <i>Externe Daten</i> → <i>Neue Abfrage erstellen...</i> wird der Dialog <i>Datenquelle auswählen</i> von MSQuery geöffnet.</p>  <p>Auf der Registerkarte <i>Datenbanken</i> den Eintrag <i><Neue Datenquelle></i> selektieren. Mit dem Button <i>OK</i> eine Neue Datenquelle anlegen.</p>

Schritt	Vorgehen: Zugriff von Excel/MSQuery
	
2	<p>Im Dialog <i>Neue Datenquelle erstellen</i> den Namen der neuen Datenquelle angeben. Dieser muß nicht mit dem Namen der WinCC-Datenbank übereinstimmen. Als Treiber <i>Sybase SQL Anywhere 5.0</i> auswählen.</p> <p>Über den Button <i>Verbinden...</i> wird der Dialog <i>Connect to SQL Anywhere</i> geöffnet, in welchem vom Treiber benötigte Informationen einzugeben sind. Als <i>User ID</i> ist <i>dba</i> und als <i>Password</i> <i>sql</i> einzugeben. Über den Button <i>Browse</i> wird die zu bearbeitende Datenbank ausgewählt.</p> <p>Mit <i>OK</i> werden die Eingaben abgeschlossen.</p> 
3	<p>Ist für die gewählte Datenbank noch keine Datenquelle konfiguriert, erscheint die Meldung <i>Name der Datenquelle nicht gefunden und kein Standardtreiber angegeben</i>.</p> <p>Diese Meldung bestätigen und ein weiteres mal den Button <i>Verbinden...</i> betätigen. Im Dialog <i>Datenquelle auswählen</i> die Registerkarte <i>Computer-Datenquelle</i> auswählen. Die CS- sowie die Runtimedatenbank des zur Zeit</p>

Schritt	Vorgehen: Zugriff von Excel/MSQuery
	<p>laufenden WinCC-Projekts sind in der Liste der Datenquellen bereits vorhanden. Der Name dieser Datenquellen beginnt mit der Zeichenfolge <i>CC_</i> gefolgt vom Projektname. Der Name der die Runtimedatenbank representierenden Datenquelle endet mit den Zeichen <i>R</i>.</p> <p>Soll jedoch eine beliebige WinCC-Datenbank bearbeitet werden, ist für diese erst eine Datenquelle anzulegen. Dies erfolgt über den Button <i>Neu</i>. Im erscheinenden Assistenten <i>Neue Datenquelle erstellen</i> auf der ersten Seite den Punkt <i>Benutzer-Datenquelle</i> auswählen und die Seite mit <i>Weiter</i> abschließen. Auf der nächsten Seite den Treiber <i>Sybase SQL Anywhere 5.0</i> auswählen und diese mit <i>Weiter</i> abschließen. Die letzte Seite mit <i>Fertigstellen</i> abschließen.</p> <p>Es wird der Dialog <i>SQL Anywhere ODBC Configuration</i> geöffnet, in welchem vom Treiber benötigte Informationen einzugeben sind. Als <i>User ID</i> ist wiederum <i>dba</i> und als <i>Password</i> <i>sql</i> einzugeben. Über den Button <i>Browse</i> wird die zu bearbeitende Datenbank ausgewählt.</p> <p>Mit <i>OK</i> wird der Dialog abgeschlossen.</p> <div data-bbox="491 779 1364 1659" style="border: 1px solid black; padding: 5px;"> <p>SQL Anywhere ODBC Configuration [X]</p> <p>Data Source Name: <input type="text" value="Example_01rt"/> <input type="button" value="OK"/></p> <p>Description: <input type="text"/></p> <p>Connection Information</p> <p>User ID: <input type="text" value="dba"/> <input type="button" value="Cancel"/></p> <p>Password: <input type="text" value="***"/> <input type="button" value="Help"/></p> <p>Server Name: <input type="text" value="<default>"/></p> <p>Database Name: <input type="text" value="Example_01rt"/></p> <p>Database Startup</p> <p>Database File: <input type="text" value="D:\Handbuch\Example_01\EXAMF"/> <input type="button" value="Browse..."/></p> <p><input type="radio"/> Local <input type="radio"/> Network <input checked="" type="radio"/> Custom <input type="button" value="Options..."/></p> <p>Additional Connection Options</p> <p>Translator Name: <input type="text" value="<No Translator>"/> <input type="button" value="Select"/></p> <p><input checked="" type="checkbox"/> Microsoft Applications (Keys in SQLStatistics)</p> <p><input checked="" type="checkbox"/> Prevent Driver not Capable errors</p> <p><input type="checkbox"/> Delay AutoCommit until statement close</p> </div> <p>Die neu erstellte Datenquelle im Dialog <i>Datenquelle auswählen</i> auswählen und den Dialog mit <i>OK</i> abschließen.</p> <p>Den nun erscheinenden Dialog <i>Connect to SQL Anywhere</i> bestätigen.</p> <p>Das Konfigurieren der Datenquelle kann schon vorher über die Systemsteuerung erfolgen. Dort den <i>ODBC Datenquellen-Administrator</i> öffnen. Über den Button <i>Hinzufügen</i> wird ebenfalls der Assistent <i>Neue Datenquelle erstellen</i> geöffnet.</p>

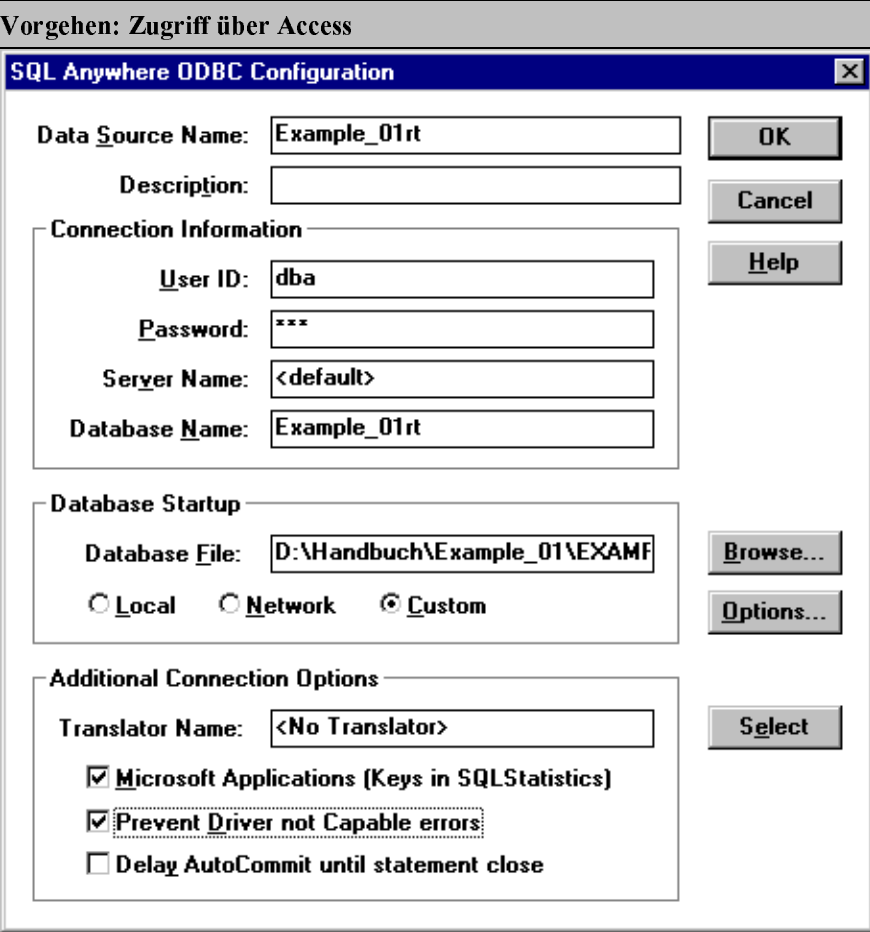
Schritt	Vorgehen: Zugriff von Excel/MSQuery
	 ODBC
4	<p>Den Dialog <i>Neue Datenquelle erstellen</i> mit <i>OK</i> abschließen.</p> <p>Im Dialog <i>Datenquelle auswählen</i> die nun neu angelegte Datenquelle selektieren und den Dialog mit <i>OK</i> abschließen.</p> <p>Auf der ersten Seite des nun erscheinenden <i>Query-Assistenten</i> werden alle verfügbaren Tabellen und Spalten angezeigt. Die gewünschten Tabellen und Spalten auswählen und die Seite mit <i>Weiter</i> abschließen. Auf den nächsten Seiten können Filter für die Daten gesetzt sowie deren Sortierreihenfolge eingestellt werden. Auf der letzten Seite wird festgelegt, ob die Daten in Excel oder in MSQuery weiterbearbeitet werden sollen. Den Dialog mit <i>Fertigstellen</i> abschließen.</p>
5	<p>Im erscheinenden Dialog <i>Externe Daten an Excel zurückgeben</i> wird die Positionierung der einzufügenden Tabellen festgelegt. Weiters können die Eigenschaften des externen Datenbereichs festgelegt werden. Der Dialog wird mit <i>OK</i> abgeschlossen.</p>

5.1.4.2 Zugriff auf die Datenbank von Access aus

Die folgende Beschreibung für den Zugriff auf die WinCC-Datenbank bezieht sich auf die Verwendung von Microsoft® Access 97 mit SR-1.

Zugriff über Access

Schritt	Vorgehen: Zugriff über Access
1	<p>Eine Access-Datenbank öffnen oder neu erstellen. Über den Menüpunkt <i>Datei</i> → <i>Externe Daten</i> → <i>Importieren...</i> wird der Dialog <i>Importieren</i> geöffnet. Als <i>Dateityp</i> den Listeneintrag <i>ODBC-Datenbanken()</i> auswählen.</p> <p>Es wird automatisch der Dialog <i>Datenquelle auswählen</i> geöffnet. Auf der Registerkarte <i>Computer-Datenquelle</i> eine Datenquelle auswählen. Die CS- sowie die Runtimedatenbank des zur Zeit laufenden WinCC-Projekts sind in der Liste der Datenquellen bereits vorhanden. Der Name dieser Datenquellen beginnt mit der Zeichenfolge <i>CC_</i> gefolgt vom Projektname. Der Name der die Runtimedatenbank representierenden Datenquelle endet mit den Zeichen <i>R</i>.</p>
2	<p>Ist die gewünschte WinCC-Datenbank noch nicht in der Liste aufgeführt, ist diese über den Button <i>Neu</i> erst als Datenquelle anzulegen.</p> <p>Im erscheinenden Assistenten <i>Neue Datenquelle erstellen</i> auf der ersten Seite den Punkt <i>Benutzer-Datenquelle</i> auswählen und die Seite mit <i>Weiter</i> abschließen. Auf der nächsten Seite den Treiber <i>Sybase SQL Anywhere 5.0</i> auswählen und diese mit <i>Weiter</i> abschließen. Die letzte Seite mit <i>Fertigstellen</i> abschließen.</p> <p>Es wird der Dialog <i>SQL Anywhere ODBC Configuration</i> geöffnet, in welchem vom Treiber benötigte Informationen einzugeben sind. Als <i>User ID</i> ist <i>dba</i> und als <i>Password sql</i> einzugeben. Über den Button <i>Browse</i> wird die zu bearbeitende Datenbank ausgewählt.</p> <p>Mit <i>OK</i> wird der Dialog abgeschlossen.</p>

Schritt	Vorgehen: Zugriff über Access
	 <p>Die neu erstellte Datenquelle im Dialog <i>Datenquelle auswählen</i> selektieren und den Dialog mit <i>OK</i> abschließen.</p>
3	Im erscheinenden Dialog <i>Objekte importieren</i> können die gewünschten Datenbanktabellen ausgewählt werden. Über <i>OK</i> werden diese in die Access-Datenbank eingefügt.

5.1.4.3 Zugriff auf die Datenbank von ISQL aus

Mit Hilfe von ISQL kann direkt auf die WinCC-Datenbank zugegriffen werden. Dies geschieht aber in Eigenverantwortung, da durch bearbeiten oder löschen von Tabellen die Projektierungsdaten inkonsistent werden können.

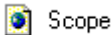
Zugriff über ISQL

Schritt	Vorgehen: Zugriff über ISQL
1	Starten von ISQL.EXE im Verzeichnis Siemens\Common\sqlany. Es erscheint der Dialog <i>Interactive SQL Logon</i> . Als <i>User ID</i> ist <i>dba</i> und als <i>Password</i> <i>sql</i> einzugeben. Wird mit <i>OK</i> bestätigt, wird das Programm automatisch mit der aktuell geöffneten WinCC-Datenbank verbunden, und zwar mit der CS-Datenbank. Soll jedoch auf eine andere Datenbank zugegriffen

Schritt	Vorgehen: Zugriff über ISQL
	werden, z.B. auf die Runtime-Datenbank, erfolgt dies über den Menüpunkt <i>Command</i> → <i>Connect</i> . Im erscheinenden Dialog sind wieder die gleichen eingaben für <i>User ID</i> und <i>Password</i> zu machen. Als <i>Database File</i> ist die gewünschte Datenbank mit kompletter Pfadangabe einzugeben.
2	<p>Im Fenster <i>Command</i> können nun SQL-Statements eingegeben werden, welche mit dem Button <i>Execute</i> ausgeführt werden.</p> <p>Es folgen einige Beispiele für SQL-Statements:</p> <ul style="list-style-type: none"> • <code>select * from systable</code> : zeigt alle Tabellennamen an • <code>select * from <tabellenname></code> : zeigt den Inhalt der Tabelle mit dem Namen <code><tabellenname></code> • <code>unload table <tabellenname> to <filename></code> : exportiert die Tabelle mit dem Namen <code><tabellenname></code> in die Datei mit dem Namen <code><filename></code> • <code>drop table <tabellenname></code> : löscht die tabelle mit dem Namen <code><tabellenname></code>

5.1.4.4 Zugriff auf die Datenbank von WinCC-Scope aus

Zugriff über WinCC-Scope

Schritt	Vorgehen: Zugriff über WinCC-Scope
1	<p>Vor dem Starten vom WinCC-Scope über das Startmenü ist die Anwendung <i>WinCCDiagAgent.exe</i> im Ordner <i>Siemens\WinCC\WinCCScope\bin</i> zu starten.</p> 
2	<p>Auf der ersten Seite über den Verweis <i>How to use the new Diagnostics Interface</i> wird eine allgemeine Beschreibung über die Bedienung von WinCC-Scope erreicht.</p> <p>Klicken Sie auf den Verweiß http://localhost, um Scope zu starten.</p>
3	<p>Im linken Bereich können aus einer Liste verschiedene Funktionen ausgewählt werden.</p> <ul style="list-style-type: none"> • Über den Punkt <i>Database</i> werden allgemeine Informationen über die WinCC-Datenbank erreicht. • Über den Punkt <i>Database Query</i> können einzelne Tabellen einer Datenbank angezeigt werden. Als <i>Data Source</i> voreingestellt ist die CS-Datenbank des aktuell geöffneten WinCC-Projects. Der Name dieser Datenquellen beginnt mit der Zeichenfolge <code>CC_</code> gefolgt vom Projektname. Der Name der die Runtimedatenbank representierenden Datenquelle endet mit den Zeichen <code>R</code>. Es können jedoch auch andere Datenquellen angezeigt werden. • Über den Punkt <i>SQL Query</i> können auf eine zu wählende Datenquelle SQL-Statements angewendet werden. Es ist jedoch ratsam, die WinCC-Datenbank nur mit sehr guten Systemkenntnissen über SQL-Statements zu bearbeiten. Beispiele für SQL-Statements sind im vorangehenden Abschnitt <i>Zugriff auf die Datenbank von ISQL aus</i> angeführt.

5.1.4.5 Export aus Datenbank über C-Aktionen

Der Datenexport kann auch aus einem WinCC-Runtime-Bild aktiviert werden. Dazu kann eine Interaktive SQL mit Kommandozeile über ProgramExecute gestartet werden. Die auszuführende Aktion ist in einer Kommandodatei (im Beispiel: archiv.sql) hinterlegt.

C-Aktion z.B. am Button

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
             lpszPropertyName)
{
    char* path = "C:\\\\SIEMENS\\\\Common\\\\SQLANY\\\\ISQL -q -b -c";
    char* parameters =
        "UID=DBA;PWD=SQL;DBN=CC_Project_97-10-21_09:53:27R";
    char* action = "read D:\\\\WinCC\\\\Project\\\\archiv.sql";

    char ExportArchive[200];
    sprintf(ExportArchive, "%s %s %s", path, parameters, action);
    ProgramExecute(ExportArchive);
}
```

- Die Variable *path* enthält den Pfad zum Programm ISQL.exe mit Aufrufparametern für dieses.
- Die Variable *parameters* enthält die im Dialog *Interactive SQL Logon* zu machenden Eingaben zur Verbindung zur Datenbank. Dies sind :
 - UID (User ID) : DBA
 - PWD (Password) : SQL
 - DBN (Data Base Name) : Name der ODBC-Datenquelle. Der Name dieser Datenquellen beginnt mit der Zeichenfolge *CC_* gefolgt vom Projektnamen und dem Datum und der Zeit der Projekterstellung. Der Name der die Runtimedatenbank representierenden Datenquelle endet mit den Zeichen *R*. Dieser Name kann bei aktiviertem Projekt über *Systemsteuerung* → *ODBC* → *Registerkarte Benutzer-DSN* ermittelt werden.
- Die Variable *action* gibt an, daß die in der Datei *archiv.sql* aufgelisteten SQL-Statements ausgeführt werden sollen.
- Die Anweisungen werden in *ExportArchive* zusammengefaßt und mit der Funktion *ProgramExecute()* ausgeführt.

Hinweis:

Soll aus einer anderen Datenbank als aus einer der zwei Projektdatenbanken ein Export durchgeführt werden, sollte statt dem Parameter DBN der Parameter DBF, das Datenbankfile mit Pfadangabe zur Datenbank, angegeben. Für die gerade aktivierte Projektdatenbank funktioniert dieser Weg jedoch nicht.

Inhalt der Datei: archiv.sql

```
select * from PDE#HD#ProcessValueArchive#Analog;  
output to D:\WinCC\Projekt\archiv.txt format ascii
```

- In der geöffneten Datenbank wird das Meßwertarchiv *pde#hd#ProcessValueArchive#Analog* selektiert und über den *output*-Befehl in die ASCII Datei *archiv.txt* exportiert.

5.1.4.6 Datenbank-Selektionen

Der zuvor beschriebene *select*-Befehl in der Kommandodatei selektiert Tabellen. Mit weiteren Parametern können Untermengen dieser Tabelle selektiert werden, die dann mit dem *output*-Befehl exportiert werden. Es folgen einige Beispiele zu diesem Thema.

Selektion auf einen Zeitbereich

```
select * from PDE#HD#ProcessValueArchive#Analog where T between  
'1996-5-1 10:10:0.00' and '1996-6-1 10:10:0.00'
```

Selektion ab einem Zeitpunkt

```
select * from PDE#HD#ProcessValueArchive#Analog where T >  
'1996-5-1 10:10:0.00'
```

Selektion auf Prozesswert ohne und mit Sortierung

```
select * from PDE#HD#ProcessValueArchive#Analog where V > 100  
select * from PDE#HD#ProcessValueArchive#Analog where V > 100  
order by T
```

Selektion mit Auswahl der Spalten T (Time) und V (Value) auf Prozesswert

```
select T,V from PDE#HD#ProcessValueArchive#Analog where V > 100  
order by T
```

5.1.5 Serielle Kopplung

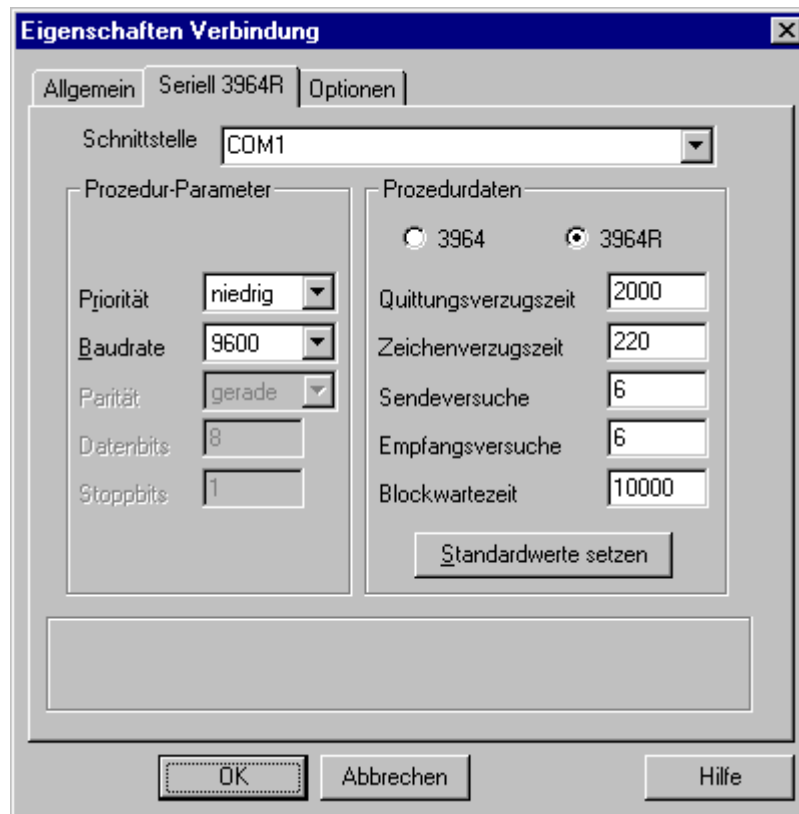
Folgende Einstellungen sind nötig, um eine serielle Kopplung aufzubauen:

Einstellungen CP525:

Nachricht:	Parameter CP525 Name:	P3964R
Prozedur:	KOMPONENTE: RK	Version : 01
Baudrate::	9600	Zeichenlänge: 8
Anzahl Stopbits: 1	Priorität: NIEDRIG	
Parität:	GERADE	

Im AG benötigt man einmal SYNCHRON im Anlaufzweig für den CP525 und SEND/RECEIVE-ALL im zyklischen Programm.

Einstellungen WinCC :



Zur Optimierung sollte einer der beiden Partner die Priorität *hoch* haben, sinnvollerweise WinCC.

5.1.6 Farbtabelle

Die Farbwerte werden aus einer großen Palette zusammengesetzt.

Die 16 Grundfarben sind:

Farbe	Farbwert (Hex)	symbolische Konstante
Rot	0x000000FF	CO_RED
Dunkelrot	0x00000080	CO_DKRED
Grün	0x0000FF00	CO_GREEN
Dunkelgrün	0x00008000	CO_DKGREEN
Blau	0x00FF0000	CO_BLUE
Dunkelblau	0x00800000	CO_DKBLUE
Cyan	0x00FFFF00	CO_CYAN
Dunkelcyan	0x00808000	CO_DKCYAN
Gelb	0x0000FFFF	CO_YELLOW
Dunkelgelb	0x00008080	CO_DKYELLOW
Magenta	0x00FF00FF	CO_MAGENTA
Dunkelmagenta	0x00800080	CO_DKMAGENTA
Hellgrau	0x00C0C0C0	CO_LTGRAY
Grau	0x00808080	CO_DKGRAY
Schwarz	0x00000000	CO_BLACK
Weiß	0x00FFFFFF	CO_WHITE

symbolische Konstante durch #define extern vordefiniert.

Mischfarben entstehen durch Zwischenwerte in der Palette.

Werden Farbänderungen mit Hilfe des Dynamik-Dialoges erstellt und die projizierten Daten mit den C-Aktionen weiterverarbeitet, können ebenfalls die Farbwerte ausgelesen werden, diese sind allerdings im dezimalen Format.

5.2 Dokumentation des S5-Meldesystems

Aufgabe und Funktion des S5-Meldesystems

Die vorliegende Dokumentation beschreibt die Funktionen und Eigenschaften der SIMATIC S5 Software:

S5-Meldesystem

Die Software dient zum zeitfolgerichtigen Erfassen von binären Meldungen, sowie deren Verarbeitung und Zwischenspeicherung. Das Programmpaket stellt innerhalb der SIMATIC S5 die notwendige Softwarefunktionalität zur Verfügung, um die Funktion 'zeitfolgerichtige Meldungserfassung' des WinCC-Systems zu realisieren.

Die prinzipielle Funktionsweise der Software läßt sich wie folgt darstellen: Die Software überwacht den binären Signalzustand von Meldungen, die der Anwender in einer Meldungsschnittstelle dem S5-Meldesystem zur Verfügung stellt. Tritt ein Wechsel in einem Signalzustand auf, so wird die Meldung mit Hilfe der Meldungsnummer identifiziert und mit dem aktuellen Datum/Uhrzeit Stempel versehen. Diese Daten werden (falls vom Anwender projektiert) um eine 32-Bit Prozeßvariable und eine alphanumerische Auftrags-/Chargenbezeichnung ergänzt. Der so gebildete Meldungsblock wird, falls notwendig, in einem Ringpuffer zwischengespeichert. Eine Zwischenspeicherung der Meldungsdaten wird immer dann benötigt, wenn mehr Meldungen pro Zeiteinheit auftreten, als über eine vorhandene BUS-Kopplung an das WinCC-System übertragen werden können. Durch diese Funktionalität wird eine zeitliche Entkopplung zwischen zeitfolgerichtiger Meldungserfassung in der SIMATIC S5 und übergeordnetem WinCC-Meldesystem erreicht und eine echtzeitfähige Meldungsverarbeitung ermöglicht.

Die vom S5-Meldesystem erzeugten Meldungsböcke werden dem S5-Anwenderprogramm in einer Datenbaustein-Schnittstelle zur Verfügung gestellt. Durch eine vom Anwender zu realisierende S5-Kopplungssoftware werden diese Daten über eine BUS-Verbindung (z.B. SINEC H1) an das übergeordnete WinCC-Meldesystem übertragen. Dort stehen für die Meldungen umfangreiche Verarbeitungsfunktionen wie z.B. Visualisierung, Archivierung, Protokollierung, usw. zur Verfügung.

Die Projektierung des S5-Meldesystems durch den Anwender erfolgt über eine Datenbausteinschnittstelle (System-DB 80). Hier legt der Anwender den Systemrahmen fest, innerhalb der das Meldesystem arbeitet. Festlegungen der vom S5-Meldesystem benutzten Speicherbereiche, Art und Umfang der zu verarbeitenden Meldungen, sowie Einteilung der belegten Adressbereiche werden hier angegeben.

Das vorliegende Kapitel beschreibt den Einsatz und die Handhabung des S5-Meldesystems in der SIMATIC S5-Umgebung. Der Anwender erhält eine Übersicht über die von der Software verwendeten Funktions- und Datenbausteine, sowie den benötigten Speicherplatz. Es folgt eine detaillierte Schnittstellenbeschreibung aller vorhandener Datenschnittstellen zwischen dem S5-Meldesystem und dem S5-Anwenderprogramm. Als Hilfestellung ist ein Projektierungsbeispiel aufgeführt, das dem Anwender den Einstieg in die Handhabung des S5-Meldesystems erleichtern soll.

5.2.1 Auflistung der Softwarebausteine

Die SIMATIC S5 Software befindet sich auf der CD mit den Beispielen zu diesem Handbuch und ist unter dem Dateinamen 'WINCC1ST.S5D' abgelegt.
Die Datei beinhaltet folgende Funktions- und Datenbausteine für das S5-Meldesystem:

FB	Name	Größe	Funktion
FB 80	SYSTEMFB	1114	Zeitfolgerichtiges Melden
FB 81	ANLAUFFB	135	Anlauf und Initialisierung für Zeitfolgerichtiges Melden
FB82	PCHECK	574	Aufgerufen durch FB 81
FB 83	MBLOCK	699	Aufgerufen durch FB 80
FB 84	SCHREIB	94	Aufgerufen durch FB 80
FB 87	VOLL	87	Aufgerufen durch FB 80
DB 80	System DB	512	Parametrieren des Meldesystems
Gesamt		2703	

Tabelle 1

Der Mindestspeicherbedarf hängt von der Projektierung des S5-Meldesystems ab. Folgende Datenbausteine werden immer zusätzlich benötigt.

Ringpuffer (min.) 2 DB = 1024 Byte
Übergabefach zu WinCC 1 DB = 512 Byte

Für jeden Offset- bzw. Parameter Datenbaustein sind weitere 512 Byte einzurechnen.

Die Berechnung der genauen Größe der Offset Datenbausteine folgt in Kapitel 5.2.4.1 Aufbau des Offset Datenbausteins, entsprechend in Kapitel 5.2.4.10 Aufbau des Parameter Datenbausteins - die Größe des jeweiligen Parameter Datenbausteins.

5.2.2 Hardwarevoraussetzung

Die in Tabelle 1 für das S5 Meldesystem angegebenen Funktionsbausteine benötigen zur korrekten Ausführung folgende Hardware:

AG	CPU
AG 115U	CPU 944 * , CPU 945
AG 135U	CPU 928B
AG 155U	CPU 946/ 947, CPU 948

Tabelle 2

* nur die CPU 944 mit zwei PG-Schnittstellen besitzt eine Systemuhr

Diese CPUs verfügen über eine interne Uhr und sind somit in der Lage, die aktuelle Datum/Uhrzeit für die Bildung der Meldungsblöcke zur Verfügung zu stellen. Über jeden eingerichteten WinCC Kanal wird ein aktuelles Datum/Uhrzeit-Telegramm zyklisch in die SIMATIC S5 CPU geschrieben. Durch den Funktionsbaustein **FB 86 : MELD:UHR** wird die interne Uhr der SIMATIC S5 mit der Uhr des WinCC-Systems synchronisiert.

5.2.3 Einbinden des S5-Meldesystems in das SIMATIC-S5 Anwenderprogramm

Um die SIMATIC S5 Software für das Meldesystem in das SIMATIC S5 Anwenderprogramm einzubinden, müssen folgende Schritte ausgeführt werden:

Alle in der Tabelle 1 angegebenen Bausteine sind von der Datei **WinCCST.S5D** auf das entsprechende AG zu übertragen.

Falls noch nicht standardmäßig implementiert, bzw. noch nicht im AG vorhanden, sind die Hantierungsbausteine für das entsprechende AG zu übertragen.

Schritt	Vorgehen: Meldesystem einbinden
1	Alle in der Tabelle 1 angegebenen Bausteine sind von der Datei WinCCST.S5D auf das entsprechende AG zu übertragen.
2	Falls noch nicht standardmäßig implementiert, bzw. noch nicht im AG vorhanden, sind die Hantierungsbausteine für das entsprechende AG zu übertragen.
3	Datenbaustein DB 80 gemäß Kapite5.2.6 parametrieren.
4	Datenbausteine für das Sendefach, den Ringpuffer, den Meldungs-Offset und ggf. die Meldungs-Parameter einrichten (siehe Kapitel 5.2.4).
5	Offset Datenbausteine für die verschiedenen Meldungsorten initialisieren (siehe Kapitel 5.2.4.1- Ruhezustand, Basismeldungsummer, ...).
6	Prozeßvariablen, Auftrags- und Chargenbezeichnung für die einzelnen Meldungen im Anwenderprogramm vorgeben (siehe Kapitel 5.2.4.10).
7	In den Anlauf-OBs (OB 20, OB 21, OB 22) sind folgende Bausteine aufzurufen: <ul style="list-style-type: none"> • SPA HTB : SYNCHRON (Der Hantierungsbaustein der jeweiligen CPU) • SPA FB 81 : ANLAUFFB
8	Im OB 1 sind folgende Bausteine aufzurufen: <ul style="list-style-type: none"> • für die zyklische Bearbeitung der Meldungen SPA FB 80 : SYSTEMFB • ein vom Anwender erstellter Funktionsbaustein zum Übertragen der Meldungsblöcke an das überlagerte WinCC-System (siehe Kapitel 5.2.5.4)
9	Weitere Funktionalitäten sind gemäß folgender Kapitel einzufügen: <ul style="list-style-type: none"> • Die Synchronisation von Datum und Uhrzeit durch FB 86 : MELD:UHR (siehe Kapitel 5.2.8).

Tabelle 3

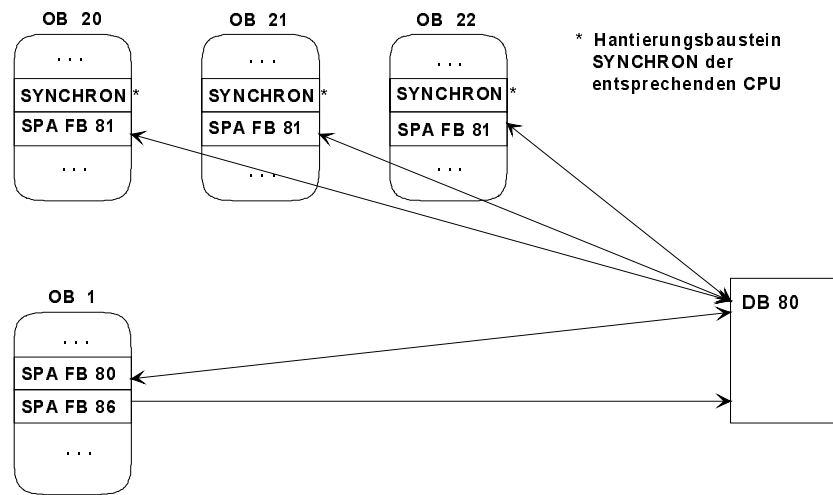


Abbildung 1

5.2.4 Allgemeine Beschreibung des S5-Meldesystems

Folgende Komponenten des S5 - Medesystems werden beschrieben:

- Offset-Datenbaustein
- Parameter-Datenbaustein
- Meldungsblock
- Ringpuffer
- Sendefach
- Systemdatenbaustein

Zusammenhang zwischen den einzelnen Komponenten:

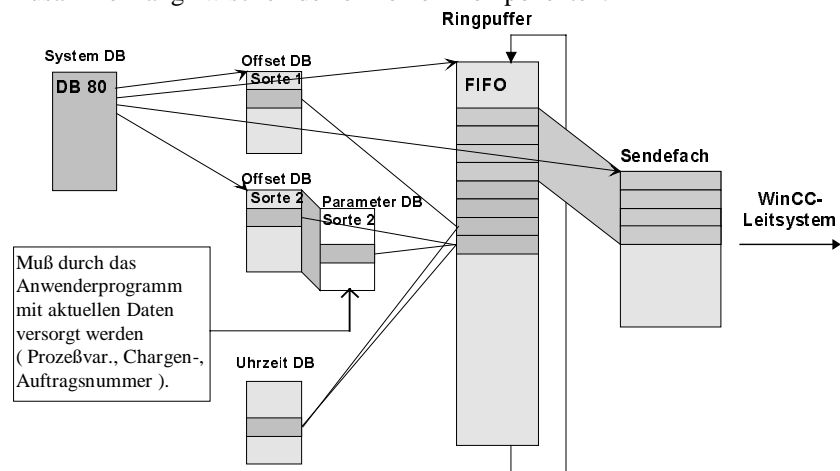


Abbildung 2

Bevor das Meldungserfassungssystem Meldungen überwachen und erfassen kann, müssen diese in den entsprechenden Datenbausteinen projektiert werden. Es werden vier Meldungssorten unterschieden:

Sorte	Definition: Meldesorten
1	Meldung ohne Parameter
2	Meldung mit Prozeßvariablen (2 DW)
3	Meldung mit Prozeßvariablen (2 DW) und Auftrags-/Chargenbezeichnung (3 DW)
4	Meldung mit Prozeßvariablen (2 DW) und Auftrags-/Chargenbezeichnung (3 DW) und Reserve (3 DW)

Tabelle 4

Für das Meldesystem, kann für die Bildung von Meldeblöcken ein Datum-/Uhrzeit-Stempel global mit angegeben werden. Bei fehlendem Datum/Uhrzeit Stempel ergänzt das WinCC-System die Meldungsblöcke (siehe Kapitel 5.2.4.11) mit den entsprechenden Informationen.

5.2.4.1 Aufbau des Offset Datenbausteins

Der Offset Datenbaustein ist für alle vier Meldungssorten gleich aufgebaut. Die entsprechende Datenbaustein-Adresse wird für jede benötigte Meldungssorte (siehe 5.2.4.1) im System Datenbaustein DB 80 angegeben.

Offset Datenbaustein für die entsprechende Meldungssorte:

DW	Inhalt	Zuordnung
DW 0	Frei	Kopf
DW 1	Basismeldungsnummer	
DW 2	Adresse des letzten Signalzustandsblocks	
DW 3	Frei	
DW 4	Signalzustände der Meldungen - Bit Nr.: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	Signalzustandsblock 1
DW 5	Ruhezustandsbits	
DW 6	Quittierungsbits	
DW 7	Flankenmerker	
DW 8	Signalzustände der Meldungen - Bit Nr.: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	Signalzustandsblock 2
DW 9	Ruhezustandsbits	
DW 10	Quittierungsbits	
DW 11	Flankenmerker	
DW 12	Signalzustände der Meldungen - Bit Nr.: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	Signalzustandsblock 3
DW 13	Ruhezustandsbits	

Tabelle 5

Nachfolgend wird beschrieben:

- Basismeldungsnummer
- Offsetmeldungsnummer
- Signalzustandsblock
- Adresse des letzten Signalzustandsblocks
- Signalzustände
- Ruhezustandsbits
- Quittierungsbits
- Flankenmerker

5.2.4.2 Basismeldungsnummer

Jeder Meldung ist eine bestimmte Meldungsnummer zugewiesen, mit der man die auftretenden Meldungen unterscheiden kann. Die Meldungsnummer setzt sich aus der Basismeldungsnummer und einer Offsetmeldungsnummer zusammen.

Für jede benutzte Meldungsorte muß eine unterschiedliche Basismeldungsnummer angegeben werden. Fortlaufend von dieser Basismeldungsnummer, werden die Meldungen dieser Sorte durch die Offsetmeldungsnummer unterschieden.

Die Basismeldungsnummer für die entsprechende Meldungsorte wird in DW 1 des zugehörigen Offset Datenbausteins angegeben (siehe 5.2.4.10).

Sonderfall

Bei Benutzung der Meldungsorte 1 ist es möglich zwei Offset Datenbausteine zu verwenden. Um eine fortlaufende Meldungsnummerierung dieser Meldungsorte zu erreichen, ist die Basismeldungsnummer des zweiten Offset Datenbausteins auf die Basismeldungsnummer des ersten Offset Datenbausteins plus dessen Meldungskapazität (1008 Meldungen) einzugeben.

Berechnung der Meldungsnummer:

Meldungsnummer = Basismeldungsnummer + Offsetmeldungsnummer

Beispiel:

Berechnung	Beschreibung:
Gegeben:	Meldungsorte 1, fortlaufende Meldungsnummerierung Beginn : Meldungsnummer 10000
Gesucht:	Basismeldungsnummer der beiden Offset Datenbausteine
10000	Basismeldungsnummer des ersten Offset Datenbausteins:
$10000 + 1008 =$ 11008	Basismeldungsnummer des zweiten Offset Datenbausteins:

5.2.4.3 Offsetmeldungsnummer / Signalzustände der Meldungen

Die Signalzustände der Meldungen befinden sich in den Offset Datenbausteinen der entsprechenden Meldungsorte an der jeweiligen Bitposition der Offsetmeldungsnummer. Die Offsetmeldungsnummer der entsprechenden Meldung ergibt sich beginnend mit den 16 Bits (Bit 0-15) von DW 4. Die fortlaufende Nummerierung erfolgt in '4-er' Schritten (DW8, DW12, ...).

Signalzustandsblock	Signalzustandsblock beginnt bei Datenwort	Bitnummer 0 - 15 entspricht Offsetmeldungsnummer
1	4	0 - 15
2	8	16 - 31
3	12	32 - 47
4	16	48 - 63
...		...
62	248	976 - 991
63	252	992 - 1007

Tabelle 6

Berechnung der Offsetmeldungsnummer:

$$\begin{aligned} \text{Offsetmeldungsnummer} &= \text{Meldungsnummer} - \text{Basismeldungsnummer} \\ \text{Offsetmeldungsnummer} &= (\text{Datenwort} / 4 - 1) * 16 + \text{Bit Nr. (0-15)} \\ \text{Offsetmeldungsnummer} &= (\text{Signalzustandsblock} - 1) * 16 + \text{Bit Nr. (0-15)} \end{aligned}$$

Berechnung von DB, DW, Bit Nr. aus der Offsetmeldungsnummer:

$$\begin{aligned} \text{Datenbaustein} &= \text{Offset Datenbaustein} \\ \text{Datenwort} &= (\text{Offsetmeldungsnummer} / 16 + 1) * 4 \\ \text{Bit Nr.} &= \text{Offsetmeldungsnummer} \% 16 \end{aligned}$$

Bei Meldungsorte 1 kann sich ein Datenwort größer 252 ergeben, dann gilt:

$$\begin{aligned} \text{Datenbaustein} &= \text{Offset Datenbaustein} + 1 \\ \text{Datenwort} &= \text{Datenwort} - 252 \\ \text{Bit Nr.} &= \text{Bit Nr.} \end{aligned}$$

Beispiel 1:

$$\begin{aligned} \text{Gegeben:} & \quad \text{DW 248, Bit 7, Basismeldungsnummer} = 10000 \\ \text{Gesucht:} & \quad \text{Meldungsnummer} \\ \\ \text{Signalzustandsblock} &= 248 / 4 \\ &= 62 \\ \text{Offsetmeldungsnummer} &= (\text{Signalzustandsblock} - 1) * 16 + \text{Bitnummer} \\ &= (62 - 1) * 16 + 7 = 983 \\ \text{Meldungsnummer} &= \text{Basismeldungsnummer} + \text{Offsetmeldungsnummer} \\ &= 10000 + 983 = 10983 \end{aligned}$$

Die gewünschte Meldungsnummer ist 10983.

Beispiel 2:

$$\begin{aligned} \text{Gegeben:} & \quad \text{Meldungsorte 1 mit zwei Offset Datenbausteinen,} \\ & \quad \text{Meldungsnummer} = 12000, \text{Basismeldungsnummer} = 10000 \\ \text{Gesucht:} & \quad \text{DB, DW, Bit Nr.} \\ \\ \text{Offsetmeldungsnummer} &= \text{Meldungsnummer} - \text{Basismeldungsnummer} \\ &= 12000 - 10000 = 2000 \\ \text{Bit Nr.} &= \text{Offsetmeldungsnummer} \% 16 = 0 \\ \text{Datenwort} &= (\text{Offsetmeldungsnummer} / 16 + 1) * 4 \\ &= (2000 / 16 + 1) * 4 = 504 \end{aligned}$$

Das Datenwort ist größer 252.

$$\begin{aligned} \text{Datenbaustein} &= \text{Offset Datenbaustein} + 1 \\ \text{Datenwort} &= 504 - 252 = 252 \\ \text{Bit Nr.} &= 0 \end{aligned}$$

Die Meldungsnummer 12000 ist im zweiten Offset Datenbaustein der 1. Meldungsorte, in Datenwort 252 Bit Nr. 0 zu finden.

5.2.4.4 Signalzustandsblock

Der erste Signalzustandsblock beginnt bei Datenwortadresse 4, Die nachfolgenden Signalzustandsblöcke folgen in einem Abstand von 4 Datenworten (DW 8, DW 12, ...). Siehe hierzu auch Tabelle 5 bzw. Tabelle 6.

Für jeden Offset Datenbaustein sind 63 Signalzustandsblöcke möglich (Signalzustandsblock 1 bis 63).

Ein Signalzustandsblock beinhaltet 16 Signalzustände. Somit ergeben sich $63 * 16 = 1008$ mögliche Meldungen in einem Offset Datenbaustein.

Aufbau des Signalzustandsblocks:

DW	Bitnummer	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	Signalzustände	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	Ruhezustände	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	Quitierungsbits	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	Flankenmarker	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabelle 7

Weitere Informationen über diese vier Bitzustände folgen in diesem Kapitel.

Berechnung des jeweiligen Signalzustandsblocks:

$$\begin{aligned} \text{Signalzustandsblock} &= (\text{Offsetmeldungsnummer} / 16) + 1 \\ \text{Signalzustandsblock} &= \text{Datenwort} / 4 \end{aligned}$$

Berechnung des Datenwortes mit dem der jeweilige Signalzustandsblock beginnt:

$$\text{Erstes Datenwort des Signalzustandsblocks} = \text{Signalzustandsblock} * 4$$

5.2.4.5 Adresse des letzten Signalzustandsblocks

Durch die Angabe der DW-Adresse, des letzten mit Meldungen belegten Signalzustandsblocks, wird die Anzahl der möglichen Meldungen der entsprechenden Meldungssorte angegeben.

Berechnung des letzten Signalzustandsblocks:

$$\text{letzter Signalzustandsblock} = \text{benötigte Meldungen dieser Meldungssorte} / 16$$

```
// nicht vollständig gefüllter (16 Meldungen) Signalzustandsblock
if ((benötigte Meldungen dieser Meldungssorte % 16) != 0)
{
    ++ letzter Signalmeldungsblock;
}
```

Bei Meldungssorte 1 kann ein Meldungsvolumen größer 1008 Meldungen auftreten, dann gilt:

1. Offset DB:

$$\begin{aligned} \text{letzter Signalzustandsblock} &= 63 \\ \text{Adresse des letzten Signalzustandsblocks} &= 63 * 4 = 252 \end{aligned}$$

2. Offset DB:

$$\text{letzter Signalzustandsblock} = \frac{(\text{benötigte Meldungen dieser Meldungssorte} - 1008)}{16}$$

```
// nicht vollständig gefüllter (16 Meldungen) Signalzustandsblock
if (((benötigte Meldungen dieser Meldungssorte - 1008) % 16) != 0)
{
    ++ letzter Signalmeldungsblock;
}
```

Berechnung der DW-Adresse des letzten Signalzustandsblocks:

$$\begin{aligned} \text{DW-Adresse des} \\ \text{letzten Signalzustandsblocks} &= \text{letzter Signalzustandsblock} * 4 \end{aligned}$$

Beispiel:

Gegeben: 1030 Meldungen der Meldungssorte 1

1. Offset DB:

$$\text{Adresse des letzten Signalzustandsblocks} = 63 * 4 = 252$$

2. Offset DB:

$$\begin{aligned} \text{benötigte Meldungen} - 1008 &= 1030 - 1008 = 22 \\ (\text{benötigte Meldungen} - 1008) / 16 &= 22 / 16 = 1 \\ (\text{benötigte Meldungen} - 1008) \% 16 &= 22 \% 16 = 6 \\ \text{Letzter Signalzustandsblock} &= 2 \\ \text{Adresse des letzten Signalzustandsblocks} &= 2 * 4 = 8 \end{aligned}$$

5.2.4.6 Signalzustände

Position: 1. Datenwort des Signalzustandsblocks (siehe Tabelle 5).

Der Anwender hat dafür zu sorgen, daß die Signalzustände der entsprechenden Meldungen in den dafür vorgesehenen Datenwörtern der Offset Datenbausteine der entsprechenden Meldungssorte eingetragen werden. Dies kann durch kontinuierlich prozeßbegleitende Signal-Aktualisierung durch das Steuerungsprogramm geschehen.

5.2.4.7 Ruhezustände

Position: 2. Datenwort des Signalzustandsblocks (siehe Tabelle 5).

Unter dem Ruhezustand eines Signals versteht man den Signalpegel im passiven Betriebszustand. Damit wird festgelegt, ob ein Signal (Meldung) 'low'- oder 'high'-aktiv arbeitet. Diese Information wird benötigt, um herauszufinden, ob eine Meldung 'kommend' oder 'gehend' ist.

Besitzt eine Ereignisänderung den negierten Zustand bezüglich des Ruhezustandes, handelt es sich um eine 'kommende' Meldung. Bei einer 'gehenden' Meldung gleicht der Zustand der Ereignisänderung dem des zugehörigen Ruhezustandes.

Die Ruhezustände der Meldungen müssen vom Anwender an den entsprechenden Positionen angegeben werden.

5.2.4.8 Quittierungsbits

Position: 3. Datenwort des Signalzustandsblocks (siehe Tabelle 5).

Quittierungsbits werden nicht projiziert, sondern innerhalb des laufenden Programms ausgewertet. Hierbei werden die Meldungen direkt von dem überlagerten PC entsprechend der jeweiligen Quittierungsphilosophie quittiert. Diese meldungsbezogenen Quittierungen sendet der PC an das betreffende AG mit den dazu projizierten Meldungen des integrierten Meldungserfassungssystems.

Das entsprechende Quittierungsbit wird vom S5 Meldesystem einen AG Zyklus lang gesetzt.

Das Anwendungsprogramm muß diese Information entsprechend auswerten.

5.2.4.9 Flankenmerker

Position: 4. Datenwort des Signalzustandsblocks (siehe Tabelle 5).

Die Flankenmerker dienen zur Feststellung aufgetretener Ereignisänderungen (Meldungsänderung). Sie werden nicht projiziert sondern innerhalb des S5-Meldesystems ausgewertet.

5.2.4.10 Aufbau des Parameter Datenbausteins

Für die Meldungsorte 2 bis 4 müssen neben einem sogenannten Offset Datenbaustein noch Parameter Datenbausteine für zusätzliche Daten der jeweiligen Meldung projiziert werden. Der Signalzustand einer Meldung ist im Offset Datenbaustein untergebracht. Die Adressen der Parameter Datenbausteine sind in fortlaufenden Datenbausteinen abgelegt und schließen sich direkt an den zugehörigen Offset Datenbaustein an.

Berechnung der Anfangsadresse eines Parameterblocks:

$$\begin{aligned} \text{Offsetmeldungsnummer} &= \text{Meldungsnummer} - \text{Basismeldungsnummer} \\ \text{Parameter DB} &= \text{Offset DB} + 1 + (\text{Offsetmeldungsnummer} / \\ &\quad \text{Parameterblöcke je Parameter DB}) \\ \text{Anfangsadresse Parameter DB} &= (\text{Offsetmeldungsnummer} \% \text{Parameterblöcke je} \\ &\quad \text{Parameter DB}) * \text{Größe Parameterblock} \end{aligned}$$

Der Anwender hat dafür zu sorgen, daß die entsprechenden Daten (Prozeßvariablen, Auftragsnummer, Chargenbezeichnung) an der entsprechenden Adresse zur Verfügung stehen.

5.2.4.11 Aufbau eines Meldungsblocks

Ein Meldungsblock, der an das überlagerte **WinCC**-System gesendet wird, besteht aus mehreren aneinander gereihten Datenwörtern. Diese enthalten alle meldungsspezifische Informationen. Die Summe der Datenwörter ergeben einen Meldungsblock. Die Größe der Meldungsblöcke differiert zwischen den einzelnen Meldungssorten (siehe Tabelle 10).

Ein Meldungsblock besteht unabhängig von der Meldungssorte immer aus mindestens zwei Datenwörtern. Dies sind die Meldungsnummer und der Meldungsstatus. Abhängig davon, ob die Meldungen mit Datum und Uhrzeit (3 Datenwörter) und entsprechenden Parametern versehen werden, kann ein Meldungsblock die maximale Größe von 12 Datenwörtern besitzen.

DW	Beschreibung:
1.DW	Meldungsnummer
2. DW	Meldungsstatus
3. DW	Uhrzeit
4. DW	Uhrzeit
5. DW	Datum
6. DW	Prozeßvariable
7. DW	Prozeßvariable
8. DW	Auftragsnummer
9. DW	Auftragsnummer
10.DW	Chargenbezeichnung
11. DW	Reserve
12. DW	Reserve

Tabelle 9

Werden die Meldungen ohne Datum/Uhrzeit Stempel versehen, entfallen die dafür nötigen drei Datenwörter an der vorgesehenen dritten bis fünften Position des Blockes. Die Parameterdatenwörter werden dann lückenlos an das Status-Datenwort angehängt. Die jeweilige Größe eines Meldungsblockes (DW-Anzahl) ist je nach Meldungssorte und gewünschtem Datum/ Uhrzeit Stempel verschieden und kann Tabelle 10 entnommen werden.

Bestimmung der Meldungsblocklänge in Abhängigkeit der Meldungsorte:

Sorte	Meldungsblocklänge in DW ohne Datum und Uhrzeit	Meldungsblocklänge in DW mit Datum und Uhrzeit
1	2	5
2	4	7
3	7	10
4	9	12

Tabelle 10

5.2.4.12 Meldungsnummer

Jeder Meldung ist eine bestimmte Meldungsnummer zugewiesen, mit der sie eindeutig identifiziert werden kann.

5.2.4.13 Meldungsstatus

Der Meldungsstatus ist wie folgt aufgebaut:

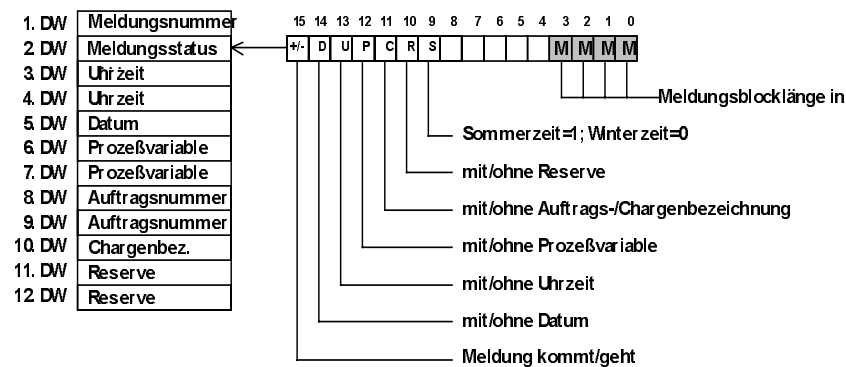


Tabelle 11

5.2.4.14 Datum-/ Uhrzeit Stempel

Das Datum und die Uhrzeit wird vom Funktionsbaustein **FB 86 : MELD:UHR** im Dual-Code zur Verfügung gestellt: Siehe Kapitel 5.2.10 Aufgabe und Funktion der S5-Uhrzeitsynchronisation.

5.2.4.15 Prozeßvariable

Zwei Datenworte, über die Prozeßvariablen bei Eintreffen einer Meldung festgehalten und an das Prozesssystem übergeben werden können.

5.2.4.16 Auftragsnummer / Chargenbezeichnung

Die ersten beiden Datenworte, sind abhängig von der Projektierung, als vorzeichenbehaftete 32-Bit Dualzahl oder als insgesamt vier ASCII-Zeichen zu interpretieren. Das dritte Datenwort ist als zwei ASCII-Zeichen zu interpretieren. Über diese drei Datenworte kann dem **WinCC**-System die aktuelle Auftragsnummer bzw. Chargenbezeichnung bei eintreffender Meldung übergeben werden.

5.2.4.17 Reserve

Die zwei Reserve-Datenworte der Meldungssorte 4 sind für zukünftige Erweiterungen vorgesehen, aber momentan noch nicht im **WinCC**-System implementiert.

5.2.4.18 Bildung eines Meldeblocks

Nachdem eine Meldung erkannt wurde, wird durch die aktuell überprüfte Bitposition die entsprechende Meldungsnummer ermittelt und als erstes Datenwort des Meldungsblockes in den Ringpuffer abgelegt. Je nach kommender oder gehender Meldung, Sorte und Wunsch auf Datum- und Uhrzeitstempel. Wird die entsprechende Statusmaske gewählt und als zweites Datenwort des Meldungsblockes in den Ringpuffer abgelegt. Ist das entsprechende Bit in dem System-Datenbaustein für einen Datum- und Uhrzeitstempel parametrieren worden, folgen nun die drei Datenwörter, welche im System Datenbaustein 80 ab Adresse DW 190 im geforderten PC-Format zur Verfügung stehen. Je nach Meldungssorte wird ggf. der zugehörige Parameterblock aus dem entsprechenden Datenarchiv (Parameter Datenbaustein) gelesen und zum Fertigstellen des Meldungsblockes in dem Ringpuffer der letzten Eingabe ergänzt. Daraufhin wird das nächste Status-Bit der folgenden Meldung untersucht. Dies wird solange durchgeführt, bis alle parametrisierten Meldungen abgearbeitet wurden.

5.2.4.19 Der interne Ringpuffer (FIFO)

Ein Ringpuffer ist ein Speicher, an dessen Ende wieder sein Anfang folgt, d.h. der Speicherbereich wird bildhaft zu einem Ring geschlossen. Damit wird erreicht, daß der Speicher einerseits von seiner Größe begrenzt wird und andererseits durch seinen jeweiligen Neubeginn an dessen Anfang nicht endlich ist. Im Meldungserfassungssystem hat dies zur Folge, daß bei dem Erreichen des virtuellen Endes ohne Entnahme der vorhergehenden Daten (Puffer = voll), die ältesten Daten durch die neusten überschrieben werden und somit als Information verloren gehen.

Der Ringpuffer im RAM dient, wie es der Name bereits sagt, als Puffer für die erfaßten Meldungen, bevor diese an den PC weitergeleitet werden. Im RAM besteht der Ringpuffer aus einem Speicherbereich von mindestens zwei Datenbausteinen und kann, je nach Parametrierung, im Rahmen der maximal zulässigen Datenbausteine eines Automatisierungsgerätes bzw. noch frei verfügbaren DBs des Anwenderprogramms, beliebig groß gewählt werden. Der Anwender teilt dem Meldesystem eine ihm für die Archivierung zur Verfügung stehende Anzahl von Datenbausteinen mit. Bei mehreren Datenbausteinen ist es Voraussetzung, Datenbausteine mit fortlaufenden DB-Nummern zu verwenden. Somit wird vom Anwender im System-DB als Parameter die Anfangs-DB-Nummer und die Nummer des End-DB's des Puffers angegeben. Alle Datenbausteine, die sich wertmäßig zwischen Anfangs-DB und End-DB befinden (inklusive der beiden Datenbausteine), gehören dem Puffer als Speicherplatz an.

5.2.4.20 Das Sendefach - Datenübertragung zum überlagerten WinCC-System

In den internen Ringpuffer werden vom S5 Meldesystem zunächst grundsätzlich alle Meldungseinträge eines jeden laufenden Zyklus eingeschrieben. Die Meldungseinträge (bis zu max. einem Datenbaustein Inhalt) werden nach abgeschlossener Erfassung in die Meldeschnittstelle (Sendefach), sofern diese bereit ist, Daten aufzunehmen, transferiert. Die Meldeschnittstelle in Form eines Datenbausteins dient für die Übertragungsfunktionsbausteine (STEP 5 - Hantierungsbausteine) als Datenquelle. Die Hantierungsbausteine bilden eine Schnittstelle zu dem entsprechenden Kommunikationsprozessor für den eingesetzten Prozeßbus (z.B. für SINEC-H1 Bus).

Aufbau des Sendefaches:

DW	Inhalt
DW 0	Länge des Datenblocks
DW 1	KY = [AG-Nr.], [CPU-Nr.]
DW 2	KY = [0], [Anzahl der Meldungen]
DW 3	Beginn der Nutzdaten (Meldungsblöcke)

Tabelle 12

DW 0:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Anstoß-Flanke								Länge des Datenblocks							

Tabelle 13

DW 0 des Sendefaches ist erstens durch das Bit Nr. 14 die Anstoß-Flanke eines gewünschten Auftrages zu entnehmen und zweitens aus den Bits Nr. 0-8 die Quelldaten-Länge zu ermitteln.

Da der zu übertragene Datenblock maximal 256 Datenwörter lang sein kann, mit einem Byte aber nur eine Zahl bis 255 darstellbar ist, ist die getrennte Abfrage der Bytes über die Befehle DL bzw. DR nicht möglich. Deshalb wird empfohlen, das DW 0 in einen Hilfsmerker zu transferieren. Dies bietet zugleich den Vorteil, daß das Freigabebit einzeln direkt auswertbar ist. Diese Operation kann bei der Verwendung von Datenwörtern nicht angewandt werden.

Bei erfüllter Bedingung sollte das Bit, welches als Flanke für einen einmaligen Anstoß eines Sendeauftrags dient, wieder zurückgesetzt werden. Die dann verbleibenden gesetzten Bits entsprechen sogleich der übertragenen Quelldaten-Länge und können in den Datenbereich der indirekten Parametrierung als QLAE geschrieben werden. Nach einem erfolgreich abgeschlossenen WRITE-Auftrag (SINEC-H1) an das **WinCC**-System ('fertig ohne Fehler' (FOF)) muß das DW 0 des Sendefaches mit dem Wert '0' überschrieben werden. Hiermit ist das Sendefach wieder freigegeben, und weitere Meldungsblöcke, sofern vorhanden, können vom internen Ringpuffer in das Sendefach übertragen werden.

Der WRITE-Auftrag (SINEC-H1) ist über die SEND-Direkt-Funktion zu realisieren und ist im entsprechenden Handbuch des AG nachzulesen.

5.2.5 Schnittstellenbeschreibung

Folgende Schnittstellen und Bausteine werden beschrieben:

- System Datenbaustein DB 80:
Zur Parametrierung des S5 Meldesystems.
- Offset Datenbaustein für die entsprechende Meldungsorte:
Binäre Schnittstelle der Meldungssignale zum S5-Meldesystem mit Spezifikation der Meldungseigenschaften.
- Parameter Datenbaustein für entsprechende Meldungsorte:
Zur Angabe zusätzlicher Meldungsdaten der Sorte 2 bis 4.
- Sendefach:
Übergabeschnittstelle zum **WinCC**-System.

5.2.5.1 System Datenbaustein 80

Mit Hilfe des System-Datenbausteins DB 80 können voneinander unabhängige Datenbereiche für vier Meldungssorten, einem Ringspeicher und einem Sendefach projektiert werden. Für die Projektierung sind Datenwort 0 bis 20 im DB 80 vorgesehen. Eine detaillierte Beschreibung der Datenworte 0 bis 20 ist in Kapitel 5.2.6 gegeben.

5.2.5.2 Offset Datenbaustein

Das S5 Meldesystem wertet die Signalzustände der entsprechenden Meldungen aus und bildet daraus bei Bedarf entsprechende Meldungsblöcke.

Der Anwender hat dafür zu sorgen, daß ...

- bei der Projektierung die Ruhezustände der einzelnen Meldungen angegeben werden.
- die Meldungszustände während der Laufzeit des S5 Anwenderprogramms, in die entsprechenden Signalzustandsbits eingeschrieben werden.
- bei Bedarf die entsprechenden Quittierungsbits ausgelesen und ausgewertet werden.

5.2.5.3 Parameter Datenbaustein

Bei der Meldungsorte 2 bis 4 können über den Meldungsblock zusätzliche Informationen über den aktuellen Anlagenzustand übergeben werden.

Der Anwender hat dafür zu sorgen, daß ...

- bei Eintreffen einer Meldung die gültigen Prozeßvariablen (Prozeßwert, Auftrags- und Chargennummer) in den entsprechenden Parameterblöcken stehen.

5.2.5.4 Sendefach / Übergabefach

Das Sendefach wird sobald es Meldungsblöcke enthält mit einem WRITE-Auftrag (SINEC-H1) direkt an das **WinCC**-System übermittelt (siehe Kapitel 5.2.4.20).

Der Anwender hat dafür zu sorgen, daß ...

- die entsprechenden Hantierungsbausteine der jeweiligen CPU vorhanden sind.
- bei der Projektierung des **WinCC**-Systems entsprechende Kommunikationskanäle für eine Prozeßbus-Anbindung angegeben sind.
- ein WRITE Auftrag, wie in Kapitel 5.2.4.20 beschrieben, angestoßen wird.

5.2.6 Parametrierung des S5 Meldesystems / System DB 80

Beschreibung der projektierbaren Datenwörter des System Datenbausteins DB 80:

DW	Beschreibung
0	DB-Adresse: interner FIFO-Anfang
1	DB-Adresse: interner FIFO-Ende
2	0: ohne Datum und Uhrzeit 1: mit Datum und Uhrzeit
3	DB-Offset für Meldungen der Sorte 1
4	1: einen DB-Offset der Sorte 1 2: zwei DB-Offset der Sorte 1
5	DB-Offset für Meldungen der Sorte 2
6	DB-Offset für Meldungen der Sorte 3
7	DB-Offset für Meldungen der Sorte 4
8	Reserve
9	Reserve
10	DB-Adresse: Sendefach CPU -> PC
11	1: Erfassungsoptimiert (EFOP)
12	EFOP ab n Meldungen
13	AG-Typ (115 / 135 / 155)
14	Reserve (muß 1 sein)
15	AG.Nr.: 1..255; CPU-Nr.: 1..4
16	Reserve
17	Reserve
18	Reserve
19	Reserve
20	PAFE der Plausibilitätsprüfung

Tabelle 14

DW 0, DW 1 : DB-Speicherbereich des internen Ringpuffers

Über die beiden Datenwörtern wird der interne Ringpufferbereich für Meldungen festgelegt.

Der Speicherplatz muß mindestens zwei Datenbausteine groß sein und es ist darauf zu achten, daß das FIFO-Ende größer als der FIFO-Anfang parametrierung wird.

Der Speicherbereich des Pufferspeichers ergibt sich aus dem Datenbausteinbereich, welcher durch FIFO-Anfang und FIFO-Ende eingegrenzt wird, einschließlich der beiden angegebenen Datenbausteine.

Die Wahl der Ringpuffergröße:

Bei Erreichen der Speicherkapazität des Ringpuffers werden die zeitlich ältesten Meldungen überschrieben. Die DB-Anzahl muß so groß gewählt werden, daß bei einem auftretenden Meldungsschwall keine Meldungen, bevor sie ausgelagert werden konnten, überschrieben werden. Damit dies nicht zum Tragen kommt, gilt folgende Faustformel.

Bestimmung der DB-Anzahl des Ringpuffers:

$Meldungen\ pro\ DB = (255\ DW / DB) /\ Meldungsblocklänge$
siehe Tabelle 10

$$\text{Anzahl der DB} = \frac{\text{Anzahl der Meldungen bei auftretendem Meldungsschwall}}{\text{Meldungen pro DB}}$$

Bei 'Erfassungsoptimierten Betrieb' des Meldungserfassungssystems ist es ratsam ein bis zwei Datenbausteine mehr einzukalkulieren.

DW 2 : Datum- und Uhrzeitkennung

Die Wahl, Meldungen mit einem Datum- und Uhrzeitstempel zu versehen, bezieht sich auf alle parametrisierten Meldungen. Entweder erhalten alle zu erfassenden Meldungen einen Datum- und Uhrzeit- Stempel (DW2 = 1) oder keine Meldungen (DW2 = 0). Wird DW2 = 0 gesetzt erweitert das WinCC-System die eintreffenden Meldeblöcke mit einem Datum/Uhrzeit Stempel.

DW 3, DW 4 : Offset-DB der Meldungsorte 1

Sind Meldungen der Sorte 1 (Meldungen ohne Parameter und Chargenbezeichnung) zu projektieren, dann ist in Datenwort 3 die Adresse des Offset Datenbausteins anzugeben. In den hier angegebenen Datenbausteinen müssen die Signalzustände dieser Meldungen fortlaufend vom Steuerungsprogramm eingeschrieben werden.

Sind mehr als 1008 Meldungen (maximal 2016 Meldungen) der Sorte 1 vorgesehen, wird durch eintragen der Zahl '2' im Datenwort 4 ein weiterer Datenbaustein für Meldungen der Sorte 1 freigegeben. Der zweite Datenbaustein hat automatisch, bezogen auf die Adresse in DW 3, die nächst höhere Adresse. Bei maximal 1008 Meldungen der Sorte 1 wird in das DW 4 eine '1' eingetragen.

DW 5, DW 6, DW 7 : Offset-DB der Meldungsorte 2, 3, 4

Die Datenwörter 5-7 enthalten analog zu DW 3 die jeweiligen Datenbaustein-Adressen, in dem die Signale der Meldungen abgelegt sind.

In DW 5 steht die Adresse des Datenbausteins für die Meldungsorte 2, in DW 6 entsprechend für die Meldungsorte 3 und in DW 7 für die Meldungsorte 4.

Kommt eine Meldungsorte nicht zum Einsatz, muß in dem entsprechenden DW eine '0' stehen.

Bei den angegebenen Adressen in DW 5-7 handelt es sich um sogenannte 'Offset-DBs'. Je nach Meldungsorte und Meldungszahl pro Sorte sind diesen eine entsprechende Anzahl von 'Folge-DB' zugeordnet. Diese enthalten die Parameter der Meldungen. Deshalb ist bei der Vergabe der Offset-DB-Adressen darauf zu achten, daß zwischen den vorhergehenden Offset-DB und dem anzugebenden genug Platz (Datenbausteine) für die Parameter-DB vorgesehen wird.

Je Meldungssorte 2-4 sind max. 1008 Meldungen projektierbar. Bei einer vollen Ausnutzung ergibt sich bei unterschiedlichen Sorten folglich eine andere Anzahl von 'Folge-DB' (Parameter-DB) zu den Offset-DB (siehe Tabelle 8).

DW 10 : Meldeschnittstelle zum überlagerten WinCC-System

Das Datenwort 10 muß immer parametrisiert werden, wobei es ohne Belang ist, in welcher Betriebsart das Meldeerfassungssystem arbeiten soll. In DW 10 wird die DB Adresse des Übergabefaches vergeben. Das Übergabefach dient als Schnittstelle von der SIMATIC S5 zum überlagerten WinCC-System.

DW 11 , DW 12 : Betriebswahl für Erfassungsoptimum und entsprechender Meldungsanzahl

Es sind zwei mögliche Betriebsarten vorgesehen:

- '0' in DW 11 -> '**Normalbetrieb**' des Meldungserfassungssystems
- '1' in DW 11 -> '**Erfassungsoptimum - Betrieb**' des Meldungserfassungssystems

Normalbetrieb:

Es werden so viele erfaßte Meldungen innerhalb eines Zyklus aus dem internen Puffer zum Senden ausgelagert, wie die Meldeschnittstelle aufnehmen kann, unter der Voraussetzung, daß sie bereit ist Daten aufzunehmen.

Bei einem sehr großen Meldungsaufkommen innerhalb eines Zyklus bzw. mehrerer aufeinanderfolgender Zyklen würde dieser Ablauf zu einer relativ hohen Zykluszeit führen. Diese wird umso größer, je größer die Meldungsblöcke der beteiligten Meldungssorten sind. Hierbei ist die Erfassung der Meldungsblöcke aufwendiger und länger.

Erfassungsoptimiert:

Bei den auftretenden Meldungen hat die zeitlich chronologische Erfassung vor der Sendung an den PC Priorität. Die relative Zeit zwischen den auftretenden Meldungen der Anlage untereinander steht im Vordergrund. Ob die Meldungen ein paar Millisekunden später bei dem PC ankommen oder nicht ist zweitrangig. Die Trägheit des menschlichen Auges und die Aufnahmefähigkeit des Beobachters in der Warte sind dafür ausschlaggebend.

Um bei solchen zeitkritischen Fällen die Zykluszeit des Meldungserfassungssystems herabsetzen zu können, ist die Betriebsmöglichkeit dieses Systems 'erfassungsoptimiert zu arbeiten' eingeführt worden. Die Mindestanzahl auftretender Meldungen innerhalb eines OB1 Zyklus ist in DW 12 anzugeben. Überschreitet die Meldungsanzahl diese Mindestanzahl während des aktuellen OB1 Zyklus werden die Meldungen nur erfaßt und gepuffert. Auf die Auslagerung und die anschließende Sendung an einen Kopplungspartner wird in diesem OB1 Zyklus verzichtet.

DW 15 : AG- / CPU-Nummer

Dieses Datenwort wird für die Bildung des Telegrammkopfes benötigt und bedarf der Angabe der projektbedingten AG- sowie der CPU-Nummer dieses Automatisierungsgerätes. Die CPU-Nummer ist vor allem dann von Bedeutung, wenn innerhalb eines AG's mehrere CPU's arbeiten. Nur im Zusammenhang mit dem Datenwort, welches die Kennung für Meldungen enthält, kann das überlagerte WinCC-System die gesendeten Daten als Meldung interpretieren, die meldungsspezifischen Meldetexte zuweisen und entsprechend auswerten.

Das DW 15 hat bei der Parametrierung als einziges Datenwort das S5-Datenformat 'KY'. Damit lassen sich zwei Bytes getrennt (durch ein Komma) darstellen. Das linke Byte beinhaltet die AG-Nummer, welche im Bereich 1 bis 255 liegen kann. Im rechten Byte wird die CPU-Nummer, die die Zahl 1 bis 4 betragen darf, angegeben.

Beispiel:

KY = 10,2
AG-Nummer = 10
CPU-Nummer = 2

DW 20 : Parametrierfehler

Alle im System-DB parametrierten Datenwörter werden beim Anlauf des S5-Meldesystems auf deren Plausibilität geprüft. Dabei wird jeweils zwischen Überschreitungen möglicher

Wertebereiche, Überschneidungen bzw. Mehrfachbelegung von parametrisierten Datenbausteinen und fehlenden Angaben unterschieden.

Als Ausgabeparameter in dem Format eines Datenwortes hat dieser Funktionsbaustein ein sog. PAFE-Wort (Parametrier-Fehler-Wort); dies ist ähnlich den systemspezifischen Hantierungsbausteinen. Der Status des PAFE-Wortes kann im System-DB 80 aus DW 20 entnommen werden. Das PAFE-Wort kann nach dem Programmrückprung aus dem FB 81 auf aufgetretene Fehler untersucht werden. Im Anschluß daran können entsprechende Handlungen vorgenommen werden.

Sinnvoll ist es, das Automatisierungsgerät bei einem PAFE-Wort, welches von Null verschieden ist, in den Stop-Zustand 'springen' zu lassen. Wird das PAFE-Wort unberücksichtigt gelassen, kann keine Garantie dafür gegeben werden, ob sich das Programm fehlerfrei verhält.

Die Auswertung des PAFE-Wortes

Wird das Programm bzw. AG nach Empfehlung bei auftreten eines Fehlers (PAFE-Wort $\neq 0$) in dessen Stop-Zustand gebracht, kann an Hand der Fehlernummer der Fehler gezielt analysiert und behoben werden. Nachfolgende Tabelle gibt über die Fehlerart, welche bei der Parametrierung verursacht wurde, Auskunft.

Format des PAFE-Wortes:

KY = Fehlernummer, Sammelfehlerkennung

Beispiel:

KY = 9,1

Aufgetretener Parametrierfehler mit der Nummer 9 entspricht:

Offset-DB-Adr. d. Sorte 1 ist größer als die max. zulässige DB-Adresse.

Fehler-Nr	Bedeutung
1	Anfang-DB des int. Puffers ist nicht definiert
2	Anfang-DB des int. Puffers ist Adressengleich mit dem System-DB ('80')
3	Anfang-DB-Adr. des int. Puffers ist größer als die max. zul. DB-Adresse
4	Ende-DB des int. Puffers ist Adressengleich mit dem System-DB ('80')
5	Ende-DB-Adr. ist kleiner als Anfang-DB-Adr. des int. Puffers
6	Ende-DB-Adr. des int. Puffers ist größer als die max. zul. DB-Adresse
7	Offset-DB d. Sorte 1 ist Adressengleich mit dem System-DB ('80')
8	Offset-DB-Adr. d. Sorte 1 liegt innerhalb des int. Pufferbereichs
9	Offset-DB-Adr. d. Sorte 1 ist größer als die max. zulässige DB-Adresse
10	Offset-DB der Sorte 2 ist Adressengleich mit dem System-DB ('80')
11	Offset-DB d. Sorte 2 ist Adressengleich mit dem der Sorte 1
12	Offset-DB der Sorte 2 ist Adressengleich mit dem 2. Offset-DB d. Sorte 1
13	Offset-DB-Adr. d. Sorte 2 liegt innerhalb des int. Pufferbereichs

Fehler-Nr	Bedeutung
14	Offset-DB-Adr. d. Sorte 2 ist größer als die max. zulässige DB-Adresse
15	Offset-DB d. Sorte 3 ist Adressengleich mit dem System-DB ('80')
16	Offset-DB d. Sorte 3 ist Adressengleich mit dem der Sorte 1
17	Offset-DB d. Sorte 3 ist Adressengleich mit dem 2. Offset-DB der Sorte 1
18	Offset-DB d. Sorte 3 ist Adressengleich mit dem der Sorte 2
19	Offset-DB-Adr. d. Sorte 3 liegt innerhalb des int. Pufferbereichs
20	Offset-DB-Adr. d. Sorte 3 ist größer als max. zulässige DB-Adresse
21	Offset-DB d. Sorte 4 ist Adressengleich mit dem System-DB ('80')
22	Offset-DB d. Sorte 4 ist Adressengleich mit dem der Sorte 1
23	Offset-DB d. Sorte 4 ist Adressengleich mit dem 2. Offset-DB der Sorte 1
24	Offset-DB-Adr. d. Sorte 4 liegt innerhalb des int. Pufferbereichs
25	Offset-DB-Adr. d. Sorte 4 ist größer als die max. zulässige DB-Adresse
26	Offset-DB d. Sorte 4 ist Adressengleich mit dem der Sorte 2
27	Offset-DB d. Sorte 4 ist Adressengleich mit dem der Sorte 3
28	PC-Sendefach ist Adressengleich mit dem System-DB ('80')
29	PC-Sendefach ist nicht definiert ('0')
30	PC-Sendefach-Adr. liegt innerhalb des int. Pufferbereichs
31	PC-Sendefach-Adr. ist größer als die max. zulässige DB-Adresse
32	PC-Sendefach ist Adressengleich mit dem Offset-DB der Sorte 1
33	PC-Sendefach ist Adressengleich mit dem Offset-DB der Sorte 2
34	PC-Sendefach ist Adressengleich mit dem Offset-DB der Sorte 3
35	PC-Sendefach ist Adressengleich mit dem Offset-DB der Sorte 4
36	PC-Sendefach ist Adressengleich mit dem 2. Offset-DB der Sorte 1
37	Reserve DW 9 bzw. Reserve DW 10 ungleich 0
38	Reserve DW 9 bzw. Reserve DW 10 ungleich 0
39	Reserve DW 9 bzw. Reserve DW 10 ungleich 0
40	Reserve DW 9 bzw. Reserve DW 10 ungleich 0
41	Reserve DW 9 bzw. Reserve DW 10 ungleich 0
42	Reserve DW 9 bzw. Reserve DW 10 ungleich 0
43	Reserve DW 9 bzw. Reserve DW 10 ungleich 0
44	Reserve DW 9 bzw. Reserve DW 10 ungleich 0
45	Reserve DW 9 bzw. Reserve DW 10 ungleich 0
46	Reserve DW 9 bzw. Reserve DW 10 ungleich 0
47	Meldungsanzahl f. d. Mindestgrenze der gewählten Betriebsart mit Erfassungsoptimum fehlt
48	AG-Typ ist nicht definiert
49	Reserve DW 14 ungleich 1
50	AG-Nr. für Telegrammkopf ist nicht definiert
51	CPU-Nr. für Telegrammkopf ist nicht definiert
52	CPU-Nr. ist größer als zulässig (1..4)

Tabelle 15

5.2.7 Projektierungsbeispiel für das S5 Meldesystem

Beschreibung

Das S5 Meldesystem soll für die folgenden Meldungssorten projiziert werden:

Sorte	Definition: Meldesorten
1	1200 Meldungen (von Meldungsnummer 10000 bis 11199) Meldungen 11000 bis 11199 sind 'low aktiv'
2	keine Meldungen vorgesehen
3	11 Meldungen (von Meldungsnummer 30000 bis 30010)
4	keine Meldungen vorgesehen

Alle Meldungen erhalten einen Datum/Uhrzeitstempel.
Verwendet wird eine 135U, AG Nummer 1, CPU Nummer 1.

5.2.7.1 Parametrierung DB 80

Sorte	max. Anzahl	Größe des Parameter-Blocks	Anzahl Blöcke je Parameter DB	max. Anzahl Parameter DBs
1	1008 / 2016	-	-	-
2	1008	2 DW	128	8
3	1008	5 DW	51	20
4	1008	7 DW	36	28

DB 81 wird als PC - Sendefach benutzt.

Bei einem gleichmäßigen Auftreten der vorhandenen Meldungen ergibt sich eine mittlere Meldungsbloklänge (mit Datum/Uhrzeit) von:
 $(1200 * 5 + 11 * 10) / (1200 + 11) = 5,04$

Annahme:

Das S5-Meldesystem soll einen Meldeschwall von 100 Meldungen in einem AG-Zyklus aufnehmen können und ab 30 Meldungen im 'Erfassungsoptimierten Betrieb' arbeiten.

$$5 \text{ DW/Mld.} * 100 \text{ Mld} = 500 \text{ DW}$$

$$(500 \text{ DW}) / (256 \text{ DW/DB}) = 1,95 \text{ DBs}$$

Für den Ringpuffer ergeben sich somit vier Datenbausteine, da für den 'Erfassungsoptimierten Betrieb' ein bis zwei Datenbausteine mehr gerechnet werden sollten. Der Ringpuffer beginnt bei Datenbausteinadresse 82 und somit ergibt sich eine Endadresse das Ringpuffers von DB 85.

Um eine Reserve für einen zukünftigen Ausbau des Ringpuffers vorzusehen, liegt der Offset-Datenbaustein der Sorte 1 auf DB 88 und DB 89 (mehr als 1008 Meldungen der Sorte 1).

DB 90 wird zum Offset-Datenbaustein der Meldungssorte 3. Ein Parameter-DB der Meldungssorte 3 hat ein Aufnahmevermögen von 51 Parameter-Blöcken, subtrahiert man die

benutzten 11 Blöcke ergibt sich mit nur einem Parameterbaustein (DB 91) eine Erweiterbarkeit von 40 Meldungen der Sorte 3.

DW	Beschreibung	Wert
0	DB-Adresse: interner FIFO-Anfang	82
1	DB-Adresse: interner FIFO-Ende	85
2	0: ohne Datum und Uhrzeit 1: mit Datum und Uhrzeit	1
3	DB-Offset für Meldungen der Sorte 1	88
4	1: einen DB-Offset der Sorte 1 2: zwei DB-Offset der Sorte 1	2
5	DB-Offset für Meldungen der Sorte 2	0
6	DB-Offset für Meldungen der Sorte 3	90
7	DB-Offset für Meldungen der Sorte 4	0
8	Reserve	0
9	Reserve	0
10	DB-Adresse: Sendefach CPU -> PC	81
11	1: Erfassungsoptimiert (EFOP)	1
12	EFOP ab n Meldungen	30
13	AG-Typ (115 / 135 / 155)	135
14	Reserve	1
15	AG.Nr.: 1..255; CPU-Nr.: 1..4	1, 1
16	Reserve	0
17	Reserve	0
18	Reserve	0
19	Reserve	0
20	PAFE der Plausibilitätsprüfung	0

Datenbaustein 100 wird von DW 10 bis DW 20 zur Uhrzeitsynchronisierung verwendet.
Datenbaustein 101 wird von DW 0 bis DW 255 für die Aufnahme von Kommandos verwendet.

5.2.7.2 Einrichten der Datenbausteine

Anlegen der Datenbausteine DB 81 - DB 85, DB 88 - DB 91 und DB 101 von DW 0 - DW 255.

Anlegen des Datenbausteins DB 100 von DW 0 - DW 20.

5.2.7.3 Initialisierung der Offset Datenbausteine

Meldungsorte 1

Für die Meldesorte 1 sind DB 88 und DB 89 vorgesehen. DB 88 beinhaltet die Meldungen mit den Meldenummern 10000 bis 11007, DB 89 die Meldungen mit den Meldenummern 11008 bis 11199.

Insgesamt sollen 1200 Meldungen der Sorte 1 projektiert werden.

Siehe Kapitel 5.2 Adresse des letzten Signalzustandsblocks :

Offsetmeldungsnummer = Meldungsnummer - Basismeldungsnummer = 0 bis 1199

1. Offset DB:

Adresse des letzten Signalzustandsblocks: DW 252

2. Offset DB:

Adresse des letzten Signalzustandsblocks: DW 252

$$1200 - 1008 = 192$$

$$192 / 16 = 12$$

$$192 \% 16 = 0$$

Adresse des letzten

Signalzustandsblock's im Offset Datenbaustein 2 = $12 * 4 = 48$

DB 88:

DW	Beschreibung	Wert
DW 0	frei	
DW 1	Basismeldungsnummer	10000
DW 2	Adresse des letzten DW	252
DW 3	frei	

DB 89:

DW	Beschreibung	Wert
DW 0	frei	
DW 1	Basismeldungsnummer	11018
DW 2	Adresse des letzten DW	48
DW 3	frei	

Siehe Kapitel 5.2.4.3 Offsetmeldungsnummer / Signalzustände der Meldungen:

Die Meldungen 11000 bis 11199 sind 'low aktiv'.

Position des Ruhezustandsbits der Meldungsnummer 11000:

Offsetmeldungsnummer: $11000 - 10000 = 1000$
 Anfang des Signalzustandsblocks $(\text{Offsetmeldungsnummer} / 16 + 1) * 4 =$
 $= (62 + 1) * 4 * DW 252$
 Datenwort der Ruhezustandsbits: DW 253
 Datenbit: $\text{Offsetmeldungsnummer} \% 16 = 8$
 Datenbaustein: $\text{Offset Datenbaustein} = DB 88$

Position des Ruhezustandsbits der Meldungsnummer 11199:

Offsetmeldungsnummer: $11199 - 10000 = 1199$
 Anfang des Signalzustandsblocks: $(\text{Offsetmeldungsnummer} / 16 + 1) * 4 =$
 $= (74 + 1) * 4 = 300$
 $300 - 252 = 48$
 Datenwort der Ruhezustandsbits: DW 49
 Datenbit: $\text{Offsetmeldungsnummer} \% 16 = 15$
 Datenbaustein: $\text{Offset Datenbaustein} + 1 = DB 89$

Folgende Ruhezustandsbits sind zu ändern:

DB 88:

DW 253: Datenbit 8 bis 15 auf '1' setzen

DB 89:

DW 5, DW 9, DW 13, ... DW 49 : Datenbit 0 bis 15 auf '1' setzen

Meldungsorte 3

Für die Meldesorte 3 ist DB 90 als Offset Datenbaustein mit den Meldungen 30000 bis 30010 und DB 91 als Parameter Datenbaustein vorgesehen.
 Insgesamt sollen 11 Meldungen der Sorte 3 projektiert werden.
 Siehe Kapitel 5.2.4.10 Aufbau des Parameter Datenbausteins:

$\text{OffsetmeldungsNr.} = \text{MeldungsNr.} - \text{BasismeldungsNr.} = 0 \text{ bis } 10$

Offset DB:

Adresse des letzten Signalzustandsblocks: $11 / 16 = 0$
 $11 \% 16 = 11$
 Adresse des letzten Signalzustandsblocks $= (0+1) * 4 = 4$

DB 89:

DW	Beschreibung	Wert
DW 0	frei	
DW 1	Basismeldungsnummer	30000
DW 2	Adresse des letzten DW	4
DW 3	frei	

Alle Ruhezustandsbits sind 0.

Siehe Kapitel 5.2.4.10 Aufbau des Parameter Datenbausteins

$$\text{Parameter DB} = \text{Offset DB} + 1 + (\text{Offsetmeldungsnummer} / \text{Parameterblöcke je Parameter DB})$$

Meldungsnummer 30000:

$$\text{Parameter DB} = 90 + 1 + 0 / 51 = 91$$

Meldungsnummer 30010:

$$\text{Parameter DB} = 90 + 1 + 10 / 51 = 91$$

$$\begin{aligned} \text{Anfangsadresse des} &= (\text{Offsetmeldungsnummer} \% \text{Parameterblöcke je Parameter} \\ \text{jeweiligen} &\text{DB}) * \text{Größe des Parameterblock} \\ \text{Parameterblocks} & \end{aligned}$$

Meldungsnummer 30000:

$$\text{Anfangsadresse des jeweiligen Parameterblocks} = (0 \% 51) * 5 = \text{DW}$$

Meldungsnummer 30010:

$$\text{Anfangsadresse des jeweiligen Parameterblocks} = (10 \% 51) * 5 = \text{DW 50}$$

DB 91: Parameter Datenbaustein 91 zu Offset Datenbaustein 90

MeldungsNr.	Prozeßwerte	Auftragsnummer	Chargenbezeichnung
30000	DW 0, 1	DW 2, 3	DW 4
30001	DW 5, 6	DW 7, 8	DW 9
30002	DW 10, 11	DW 12, 13	DW 14
30003	DW 15, 16	DW 17, 18	DW 19
30004	DW 20, 21	DW 22, 23	DW 24
30005	DW 25, 26	DW 27, 28	DW 29
30006	DW 30, 31	DW 32, 33	DW 34
30007	DW 35, 36	DW 37, 38	DW 39
30008	DW 40, 41	DW 42, 43	DW 44
30009	DW 45, 46	DW 47, 48	DW 49
30010	DW 50, 51	DW 52, 53	DW 54

5.2.8 Dokumentation der SIMATIC S5 Kommandobausteine

Aufgabe und Funktion der S5-Kommandobausteine

Die Software dient zum 'Bearbeiten' von Bits, Bytes, Worten und Doppelworten in der SIMATIC S5 über einen Prozeßbus (z.B. SINEC H1). Über den Prozeßbus ist es nur möglich Byte- bzw. Wortwerte in der SIMATIC S5 anzusprechen.

Folgende Operationen sind standardmäßig ausführbar:

- Datenbausteine (DB und DX), Timer und Counter sind nur als Wort zu ändern.
- Merker, Eingänge, Ausgänge, Peripherie (P und Q) sind nur als Byte zu ändern.

Das Programmpaket stellt innerhalb der SIMATIC S5 die notwendige Softwarefunktionalität zur Verfügung, um die folgende Operationen vom WinCC-System aus über den gegebenen Prozeßbus zu realisieren:

- Für einen OB1-Zyklus lang einen Richtimpuls setzen
- Bit in DB / DX setzen / rücksetzen / invertieren
- Bit in Merker setzen / rücksetzen / invertieren
- Byte links / rechts in DB / DX schreiben
- Wort / Doppelwort in DB / DX schreiben
- Byte / Wort in Merker schreiben
- Byte / Wort in Peripherie schreiben
- Byte / Wort in erweiterte Peripherie schreiben

Die gewünschten Änderungen in der SIMATIC S5 werden über eine Daten-Schnittstelle vom WinCC - Control Center als Rohdatenvariable zur Verfügung gestellt. Die Befehle müssen über die Rohdatenvariable an die S5 geschickt werden. Diese Befehle werden direkt in der S5 durch einen Befehlsinterpreter **FB 87 : EXECUTE** ausgewertet und ausgeführt.

Das vorliegende Handbuch beschreibt detailliert den Einsatz und die Handhabung der S5-Kommandobausteine in der SIMATIC S5 - Umgebung. Der Anwender erhält eine Übersicht über die von der Software verwendeten Funktions- und Datenbausteine, sowie den benötigten Speicherplatz. Es folgt eine detaillierte Schnittstellenbeschreibung der vorhandenen Datenschnittstelle. Als Hilfestellung ist ein Projektierungsbeispiel aufgeführt.

5.2.8.1 Auflistung der Softwarebausteine

Die SIMATIC S5 Software '*S5-Kommandobausteine*' befindet sich auf der WinCC - CD unter dem Dateinamen **WINCC1ST.S5D**.

Die Datei beinhaltet folgende Funktionsbausteine für die 'S5-Kommandobausteine':

FB	Name	Größe in Byte	Funktion
FB 87	EXECUTE	152	Ermöglicht Bit-, Byte-, Wort-, Doppelwort - Manipulationen über den Prozeßbus
FB 88	OPCODE	399	Aufgerufen durch FB 87
Gesamt		551	

Tabelle 16

Zusätzlich wird ein Kommando Datenbaustein mit der Größe 512 Byte benötigt.

5.2.8.2 Hardwarevoraussetzung

Die in der Tabelle 16 angegebenen Funktionsbausteine benötigen zur korrekten Ausführung folgende Hardware:

AG	CPU
AG 115U	CPU 943, CPU 944, CPU 945
AG 135U	CPU 928A, CPU 928B
AG 155U	CPU 946/ 947, CPU 948

5.2.8.3 Aufrufparameter von FB 87: EXECUTE

Im folgenden werden die Aufrufparameter von Funktionsbaustein **FB 87 : EXECUTE** beschrieben.

Name	Execute	Parameter
Bez:	DBNR	E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KF
Bez:	DBDX	E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KF
Bez:	RIMP	E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KY

DBNR: Datenbaustein-Nummer der Kommandoübergabe-Schnittstelle

DBDX: Typ der Datenquelle für die Kommandoübergabe-Schnittstelle

DB.....Datenquelle ist ein Datenbaustein (DB).

DX.....Datenquelle ist ein erweiterter Datenbaustein (DX).

RIMP: Bitposition für den Richtimpuls

RIMP.....Merker-Nummer , Bit-Nummer

5.2.9 Schnittstellenbeschreibung

Folgende Schnittstellen und Bausteine werden beschrieben:

- Kommando-Funktionsbaustein FB 87
- Kommando-Datenbaustein: Befehlsübergabeschnittstelle zur SIMATIC S5.

In der SIMATIC S5 wird im OB 1 der Befehlsinterpreter (**FB 87 : EXECUTE**) zyklisch aufgerufen. Art und Adresse des Kommando DB's werden als Parameter übergeben. Bei anstehendem Kommando wird der Opcode und vier Parameter an **FB 88 : OPCODE** weitergeleitet und direkt ausgeführt. Nach ausgeführtem Befehl wird der Kommandozähler (DW 1) um eins dekrementiert. Der Vorgang der Befehlsübergabe und des Dekrementierens des Kommandozählers wiederholt sich bis alle anstehenden Befehle abgearbeitet wurden.

Angaben über Typ und Adresse des Datenbausteins müssen im WinCC Control-Center, als auch im S5 Programm übereinstimmen und der Datenbaustein in der S5 vorhanden sein. Als Auswahl steht ein DB- bzw. ein DX-Datenbaustein und dessen Adresse (z.B. DX 234) zur Verfügung. Der Datenbaustein muß bis Datenwort 255 vom Anwender geöffnet werden, da Datenwort 0-255 im angegebenen Datenbaustein angesprochen werden können.

Folgende Syntax der abgelegten Kommandos im Kommando-Datenbaustein ist festgelegt:

DW	Beschreibung
0	nicht benutzt
1	Anzahl der auszuführenden Kommandos
2	Opcode des ersten Kommandos
3	Parameter 1 (Opcode 1)
4	Parameter 2 (Opcode 1)
5	Parameter 3 (Opcode 1)
6	Parameter 4 (Opcode 1)
7	Opcode des zweiten Kommandos
8	Parameter 1 (Opcode 2)
9	Parameter 2 (Opcode 2)
10	Parameter 3 (Opcode 2)
11	Parameter 4 (Opcode 2)
12	Parameter 1 (Opcode 2)
13	Opcode des dritten Kommandos
14	Parameter 1 (Opcode 3)
.....	

Nachfolgend sind die Implementierten Kommandos syntaktisch beschrieben:

Übergabe von Opcode und Parametern in den Kommando DB

Befehl	Opode	Param- eter 1	Param- eter 2	Param- eter 3	Param- eter 4
Bit in DB setzen	10	DB	DW	Bit	-
Bit in DB rücksetzen	11	DB	DW	Bit	-
Bit in DB invertieren	12	DB	DW	Bit	-
Byte rechts in DB setzen	15	DB	DW	Wert	-
Byte links in DB setzen	16	DB	DW	Wert	-
Datenwort in DB schreiben	17	DB	DW	Wert	-
Doppelwort in DB schreiben	18	DB	DW	Wert	Wert
Bit in DX setzen	20	DX	DW	Bit	-
Bit in DX rücksetzen	21	DX	DW	Bit	-
Bit in DX invertieren	22	DX	DW	Bit	-
Byte rechts in DX setzen	25	DX	DW	Wert	-
Byte links in DX setzen	26	DX	DW	Wert	-
Datenwort in DX schreiben	27	DX	DW	Wert	-
Doppelwort in DX schreiben	28	DX	DW	Wert	Wert
Merkerbit setzen	30	MB	Bit	-	-
Merkerbit rücksetzen	31	MB	Bit	-	-
Merkerbit invertieren	32	MB	Bit	-	-
Merkerbyte schreiben	35	MB	Wert	-	-
Merkerwort schreiben	36	MW	Wert	-	-
Peripheriebyte schreiben	45	PB	Wert	-	-
Peripheriewort schreiben	46	PW	Wert	-	-
Erweitertes Peripheriebyte schreiben	55	QB	Wert	-	-
Erweitertes Peripheriewort schreiben	56	QW	Wert	-	-
Richtimpuls setzen	60	-	-	-	-

5.2.9.1 Projektierungsbeispiel für die S5 Kommandobausteine

Die S5 Kommandobausteine sollen eingerichtet werden.
Der Richtimpuls wird in Merkerwort 56, Bit 4 zur Verfügung gestellt. Als Kommando-Datenbaustein soll DX 237 dienen. Es ist dafür zu sorgen, daß im AG der Datenbaustein DX 237 von DW 0 bis 255 geöffnet ist.
Im WinCC Control-Center ist bei der Angabe der Kanalparameter (z.B. SINEC H1) der gewünschte Datenbaustein einzutragen.

Auszug aus OB 1:

```
.....  
: SPA FB 87  
NAME : EXECUTE  
DBNR : KF +237  
DBDX : KC DX  
RIMP : KY 56, 4  
.....
```

5.2.10 Aufgabe und Funktion der S5-Uhrzeitsynchronisation

Die vorliegende Dokumentation beschreibt die Funktionen und Eigenschaften der SIMATIC S5 Software:

S5-Uhrzeitsynchronisation

Die Software dient zur Synchronisation der SIMATIC S5 Systemuhr. Weiterhin liefert sie der 'zeitfolgerichtigen Meldungserfassung' des S5-Meldesystems ein passendes Datum-/Uhrzeit-Datenformat zur Bildung der Meldungsblöcke.

Der Funktionsbaustein **FB 86 : MELD:UHR** stellt zusätzlich die aktuelle S5-Uhrzeit in einem für die 'zeitfolgerichtige Meldungserfassung' benötigten Format zur Verfügung. Die Daten liegen im System Datenbaustein 80 ab DW 190 zur Verfügung.

Tritt ein Wechsel in einem Meldungs-Signalzustand auf, so wird die Meldung mit Hilfe der Meldungsnummer vom Funktionsbaustein **FB 80 : SYSTEMFB** identifiziert und mit dem aktuellen Datum- /Uhrzeit-Stempel aus System-Datenbaustein 80 versehen.

Das vorliegende Handbuch beschreibt detailliert den Einsatz und die Handhabung der S5-Uhrzeitsynchronisation in der SIMATIC S5-Umgebung. Der Anwender erhält eine Übersicht über die von der Software verwendeten Funktions- und Datenbausteine, sowie den benötigten Speicherplatz. Als Hilfestellung ist ein Projektierungsbeispiel aufgeführt.

5.2.10.1 Auflistung der Softwarebausteine

Die SIMATIC S5 Software (S5-Uhrzeitsynchronisation) befindet sich auf der WinCC - CD unter dem Dateinamen: **WINCC1ST.S5D**.

Die Datei beinhaltet folgende Funktions- und Datenbausteine:

FB	Name	Größe in Byte	Funktion
FB 86	MELD:UHR	1135	Synchronisation der Uhrzeit
Gesamt		1135	

Tabelle 17

Uhrendatenbereich 115U: 27 DW = 54 Byte

Uhrendatenbereich 135U/155U: 12 DW = 24 Byte

Datenbereich für das S5 Meldesystem: 3 DW = 6 Byte

5.2.10.2 Hardwarevoraussetzung

Die für das S5 Meldesystem angegebenen Funktionsbausteine benötigen zur korrekten Ausführung folgende Hardware:

AG	CPU
AG 115U	CPU 944 *, CPU 945
AG 135U	CPU 928B
AG 155U	CPU 946/ 947, CPU 948

* nur die CPU 944 mit zwei PG-Schnittstellen besitzt eine Systemuhr

5.2.11 Aufrufparameter von FB 86 : MELD:UHR

Im folgenden werden die Aufrufparameter von Funktionsbaustein **FB 86 : MELD:UHR** beschrieben.

Im folgenden werden die Aufrufparameter von Funktionsbaustein **FB 87 : EXECUTE** beschrieben.

Name	MELD:UHR	Parameter
Bez:	CPUT	E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KF
Bez:	DCF7	E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KF
Bez:	QTYP	E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KF
Bez:	QSYN	E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KY
Bez:	UDAT	E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KY
Bez:	ZINT	E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KF
Bez:	ZUHR	E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KY
Bez:	ZSYN	E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KF

CPUT:

Nr. der CPU	Typ
1	CPU 943 / CPU 944
2	CPU 945
3	CPU 928B
4	CPU 946 / 947
5	CPU 948

DCF7:

Betriebsmodus

0 = Betrieb mit S5 - System - Uhr

1 = Betrieb mit DCF77 - Funkuhr

QTYP:

Typ der Datenquelle für das Uhrzeit-Synchronisations-Telegramm.

0 = Datenquelle ist ein Datenbaustein (DB)

1 = Datenquelle ist ein erweiterter Datenbaustein (DX)

QSYN:

Datenquelle der Uhrzeitdaten

DCF7 = QSYN DB-Nummer, DW-Nummer des empfangenen

0: = Uhrzeitsynchronisationstelegramm

DCF7 = QSYN DB-Nummer, DW-Nummer der DCF77 - Uhrzeit

1: =

UDAT:**Adresse des Uhrendatenbereichs**

UDAT = DB-Nummer, DW-Nummer

ZINT:

Zeitintervall in Minuten für das Senden des Synchronisations - Telegramms (DCF7 = 1)

ZUHR:**Zieldatenbereich für Uhrzeitdaten im Meldesystem - Format.**

ZUHR = DB-Nummer , DW-Nummer

ZSYN:

Zieldatenbereich für Uhrzeit-Synchronisations-Telegramm(DCF7 = 1)

Soll die Funktionalität der 'zeitfolgerichtigen Meldungserfassung' mit dem S5 Meldesystem eingesetzt werden, wird ein spezielles Uhrzeit-Datenformat in DB 80 ab DW 190 erwartet. Dieses Uhrzeit-Datenformat wird aus der S5 System-Uhrzeit gewonnen und in den entsprechenden Datenbereich ZUHR (DB 80, DW 190-192) geschrieben.

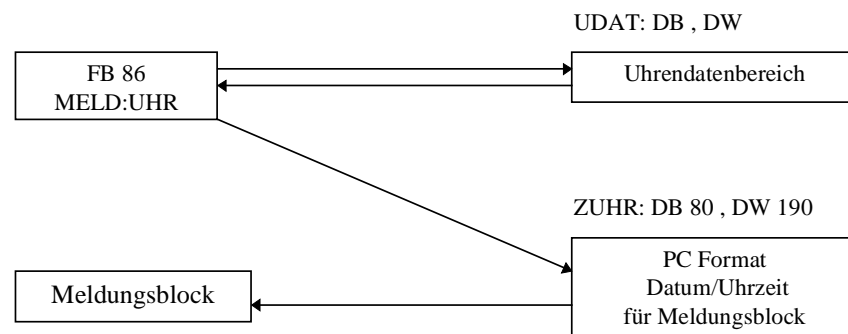
Zusammenhang zwischen 'zeitfolgerichtigem Melden' und FB 86 : MELD:UHR :

Abbildung 4

5.2.12 Datenformate für Datum und Uhrzeit

Uhrzeit - Synchronisationstelegramm von einem System aus (WinCC unterstützt zur Zeit das Uhrzeit - Telegramm nicht !!)

Im Ersten Datenwort des Uhrzeit-Synchronisations-Telegramm steht eine Quell-Kennung, die vom System mit den Datum- und den Uhrzeitdaten mitgesendet wird.

Der Funktionsbaustein **FB 86 : MELD:UHR** entnimmt das anstehende Telegramm erst, sobald an dieser Stelle die Quell-Kennung 'FFFF' ansteht. Der Empfang des Telegramms wird mit einer '0' in diesem Datenwort quittiert. Erst bei Eintreffen eines neuen Telegramms (DW 1 = 'FFFF') wird es erneut ausgelesen und ausgewertet.

Bedeutung	Datenwort	Inhalt	Gültigkeitsbereich	Bemerkung
Quell-Kennung / Uhrzeittelegramm	1	FFFF		
Telegramm - ID	2	FFFF		nicht Benutzt
Sekunden	3	00xx	xx: 0..59	
Minuten	4	00xx	xx: 0..59	
Stunden	5	00xx	xx: 0..23	
Tag	6	00xx	xx: 1..31	
Monat	7	00xx	xx: 1..12	
Jahr	8	00xx	xx: 0..127 (1990-2117)	Jahr + 1990
Tag der Woche	9	00xx	xx: 0..6	Sonntag = 0
Tag des Jahres	10	00xx	xx: 1..365	
Sommerzeit, Winterzeit Schaltjahr	11	yyxx	xx: Winterzeit = 00 Sommerzeit = 01 yy: Schaltjahr aktuelles Jahr = 00 letztes Jahr = 01 vor zwei Jahren = 02 vor drei Jahren = 03	

Tabelle 18

5.2.12.1 Uhrendatenbereich CPU 944, CPU 945

Die Datenwortnummern sind relative Angaben. Die tatsächliche Lage des Bereichs wird bestimmt durch die Aufrufparameter : UDAT = DB-Nr., DW-Nr. von **FB 86** :

MELD:UHR.

DW	Bereich
0	Interne Variablen
1	
2	
3	
4	Aktuelle Uhrzeit
5	
6	
7	
8	Uhrzeit Stellbereich
9	
10	
11	
12	Reserve - Weckzeit
13	
14	
15	
16	Reserve - Betriebsstunden
17	
18	
19	
20	
21	
22	Uhrzeit / Datum nach RUN / STOP
23	
24	
25	
26	

Tabelle 19

Aktuelle Uhrzeit im Uhrendatenbereich:

DW	Wort/Links	Wort/Rechts
4	---	Wochentag
6	Tag	Monat
7	Jahr	AM/PM (Bit, Nr.7), Stunde
8	Minute	Sekunde

Abbildung 5

Stellbereich im Uhrendatenbereich:

DW	Wort/Links	Wort/Rechts
9	Schaltjahr	Wochentag
10	Tag	Monat
11	Jahr	AM/PM (Bit, Nr.7), Stunde
12	Minute	Sekunde

Abbildung 6

5.2.12.2 Uhrendatenbereich CPU 928B, CPU 948

Die Datenwortnummern sind relative Angaben. Die tatsächliche Lage des Bereichs wird durch die Aufrufparameter : UDAT = DB-Nr., DW-Nr. von **FB 86 : MELD:UHR** bestimmt.

DW	Bereich
0	Interne Variablen
1	
2	
3	
4	Aktuelle Uhrzeit
5	
6	
7	
8	Uhrzeit Stellbereich
9	
10	
11	

Abbildung 7

Aktuelle Uhrzeit im Uhrendatenbereich:

DW	Wort/Links								Wort/Rechts						
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
4	Sekunden								0						
5	Format		Stunden						Minuten						
6	Monatstag								Wochentag			0			
7	Jahr								Sekunde						

Abbildung 8

Aktuelle Uhrzeit im Stellbereich:

DW	Wort/Links								Wort/Rechts						
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
8	Sekunden								0						
9	Format		Stunden						Minuten						
10	Monatstag								Wochentag			0			
11	Jahr								Sekunde						

Abbildung 9

5.2.12.3 Uhrendatenbereich CPU 946, CPU 947

Die Datenwortnummern sind relative Angaben. Die tatsächliche Lage des Bereichs wird durch die Aufrufparameter : UDAT = DB-Nr., DW-Nr. von **FB 86 : MELD:UHR** bestimmt.

DW	Bereich
0	Interne Variablen
1	
2	
3	
4	Aktuelle Uhrzeit
5	
6	
7	
8	Uhrzeit Stellbereich
9	
10	
11	

Abbildung 10

Aktuelle Uhrzeit im Uhrendatenbereich:

DW	Wort/Links		Wort/Rechts	
4	10er Sek.	1er Sek.	1/10 Sek.	1/100 Sek.
6	10er Std.	1er Std.	10er Min.	1er Min.
7	10er Tag	1er Tag	Wochentag	0
8	10er Jahr	1er Jahr	10er Monat	1er Monat

Abbildung 11

Aktuelle Uhrzeit im Stellbereich:

DW	Wort/Links		Wort/Rechts	
9	10er Sek.	1er Sek.	1/10 Sek.	1/100 Sek.
10	10er Std.	1er Std.	10er Min.	1er Min.
11	10er Tag	1er Tag	Wochentag	0
12	10er Jahr	1er Jahr	10er Monat	1er Monat

Abbildung 12

5.2.12.4 Uhrendatenformat für Meldungsblöcke

Die Datenwortnummern sind relative Angaben. Die tatsächliche Lage des Bereichs wird durch die Parameter : ZUHR = DB-Nr., DW-Nr. von **FB 86 : MELD:UHR** bestimmt. Soll die Funktionalität der 'Zeitfolgerichtigen Meldungserfassung' mit dem S5 Meldesystem verwendet werden, so sind in den Parameter ZUHR die Daten DB 80, DW 190 einzutragen. Das Datum und die Uhrzeit wird von Funktionsbaustein **FB 86 : MELD:UHR**, für die Meldungsverarbeitung im Dual-Code zur Verfügung gestellt:

Aktuelle Uhrzeit im Stellbereich:

Bedeutung	Datenwort	Bit	Gültigkeitsbereich	Bemerkung
1/100 Sekunden	1	0 - 6	0..99 (0 - 990 ms)	in einem 10 ms Raster
Sekunden	1	7 - 12	0..59	
Minuten	0	0 - 5	0..59	
Stunden	0	6 - 10	0..23	
Tag	2	0 - 4	1..31	
Monat	2	5 - 8	1..12	
Jahr	2	9 - 15	0..127 (1990-2117)	Jahr + 1990

Abbildung 13

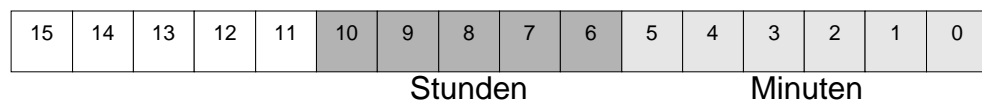
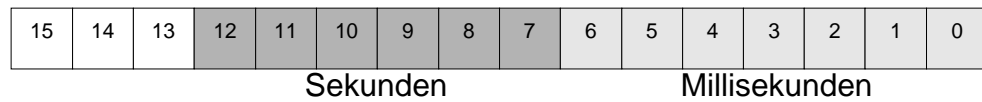
DW3: Uhrzeit**DW4: Uhrzeit****DW4: Datum**

Abbildung 14

5.2.13 Schnittstellenbeschreibung

Um die Software S5 Uhrzeitsynchronisation einzusetzen, muß der Anwender:

- Die Aufrufparameter von **FB 86 : MELD:UHR** entsprechend Kapitel 5.2.11 Aufrufparameter von FB 86 : MELD:UHR ausfüllen
- Die Datenbereiche im AG zu öffnen.

Projektierungsbeispiel

Gegeben ist eine CPU 944 mit zwei PG-Schnittstellen. Auf dieser CPU soll die S5 Uhrzeit-synchronisation für das S5 Meldesystem ohne DCF77-Uhr eingerichtet werden.

Datenbereiche:

Uhrzeit-Synchronisations-Telegramm: DB 100, DW 20 - DW 30

Uhrendatenbereich der S5 Systemuhr: DB 100, DW 31 - DW 47

Meldungsblockdaten*: DB 80, DW 190 -DW 192

* Bei Verwendung des SIMATIC S5 Meldesystems ist dieser Datenbereich fest vorgeschrieben.

Im System ist bei der Angabe der Kanalparameter (z.B. SINEC H1) der gewünschte Datenbaustein (DB 100, DW20 - DW 30) bei der Angabe 'Uhrzeit-Synchronisation:' einzutragen.

Es ist dafür zu sorgen, daß DB 80 von DW 0 bis DW 255 und DB 100 von DW 0 bis DW 47 geöffnet sind.

Auszug aus OB 1:

.....	
: SPA	FB 86
NAME	: MELD: UHR
CPUT	: KF +1
DCF7	: KF +0
QTYP	: KF +0
QSYN	: KY 100, 20
UDAT	: KY 100, 31
ZINT	: KF +0
ZUHR	: KY 80, 190
ZSYN	: KF +0
.....	

5.2.14 Zusammenspiel mit dem WinCC - Meldesystem

Folgendes ist beim Zusammenspiel des WinCC-Meldesystems mit dem S5-Meldebaustein zu beachten :

In der S5 muß im Sendebaustein als Anzahl zu übertragene Datenworte 256 angegeben werden.

Im Control-Center muß im S5Trsp-Kanal eine neue Verbindung eingerichtet werden. Beim Punkt Verbindung/Read-Funktion den Typ Fetch-passiv angeben.

Für den Datenaustausch mit dem Meldesystem müssen pro AG 2 Rohdatenvariablen angelegt werden.

Die erste ist für das Empfangen von Meldungen zuständig.

Ihre Addressierung ist wie folgt einzustellen : Datenbereich : DB, DB-Nr. : xx,

Adressierung Wort, DW : 0, Rohdatentyp : Ereignis

Die zweite wird für das Senden der Quittungsinformationen benötigt

Ihre Addressierung ist wie folgt einzustellen : Datenbereich : DB, DB-Nr. : 80,

Adressierung Wort, DW : 90, Rohdatentyp : Ereignis

Im Meldesystem die Ereignisvariable mit der Empfangsrohdatenvariable verbinden(Die Bitinformation ist hier ohne Bedeutung).

Die Quittungsvariable mit der Senderohdatenvariable verbinden(Die Bitinformation ist hier ohne Bedeutung).

Als Normierungs-DLL die S5STD.NLL - Datei eintragen.

Tip : Mit dem Verschaltungs-Wizard können alle betroffenen Meldungen auf einmal verbunden werden.

Für die Prozeßwerte sind nur positive Festpunkt-Zahlen zulässig. Gleitkommawerte werden nicht unterstützt.

S5		WinCC
Prozeßwert	➡	Prozeßwert 1
Auftragsbez.	➡	Prozeßwert 2
Chargenbez.	➡	Prozeßwert 3
Reserve	➡	Prozeßwert 4

5.3 Schnittstelle Normierungs-DLL zu AlarmLogging und TagLogging

Zielsetzung

Die Applikationen AlarmLogging und TagLogging erfassen die Prozeßdaten über den WinCC Datenmanager. Abhängig vom Kommunikationstyp zum Prozeß sind

- unterschiedliche Kanal-DLL's am Datentransfer beteiligt,
- die Prozeßdaten in Telegrammen (Rohdatenvariablen) mit unterschiedlichem Aufbau abgelegt.

AlarmLogging und TagLogging sollen die Prozeßdaten jedoch unabhängig vom jeweiligen Kommunikationstyp auf dieselbe Art und Weise verarbeiten können. Deshalb wird für jeden Kommunikationstyp eine eigene Normierungs-DLL eingesetzt, die den genauen Aufbau der jeweiligen Telegramme kennt und daraus eine für AlarmLogging und TagLogging allgemein gültige „Prozeßdatenform“ ableitet.

Eine Normierungs-DLL gehört prinzipiell zu einer Kanal-DLL, sie soll genau wie diese einfach ins Gesamtsystem hinzugefügt bzw. wieder entfernt werden können. Trotzdem hat sie keine direkte Schnittstelle zu der zugehörigen Kanal-DLL.

Diese Unterlage beschreibt die Einbindung und Schnittstelle jeder Normierungs-DLL zu den WinCC-Applikationen AlarmLogging und TagLogging. Sie ist entstanden bei dem Entwurf der S7PMC-Normierungs-DLL, deshalb ist der Begriff „S7PMC- Normierungs-DLL“ größtenteils gleichzusetzen mit dem Begriff „Normierungs-DLL“.

Prinzipieller Ablauf

Die S7PMC-Normierungs-DLL ist ein passives Programmgebilde, das ausschließlich zu den Applikationen AlarmLogging und TagLogging Schnittstellen besitzt. Die S7PMC-Normierungs-DLL bearbeitet S7PMC-spezifische Funktionen für AlarmLogging und TagLogging.

AlarmLogging und TagLogging melden sich mit einem Startaufruf bei der Normierungs-DLL an. Dabei werden bestimmte Parameter in einer Startstruktur an die Norm.DLL weitergegeben sowie deren Eigenschaften anhand von Kennungen entgegengenommen.

Zur Bearbeitung der S7PMC-Funktionen im Runtime-Modus sind zwei Datentransferrichtungen erforderlich:

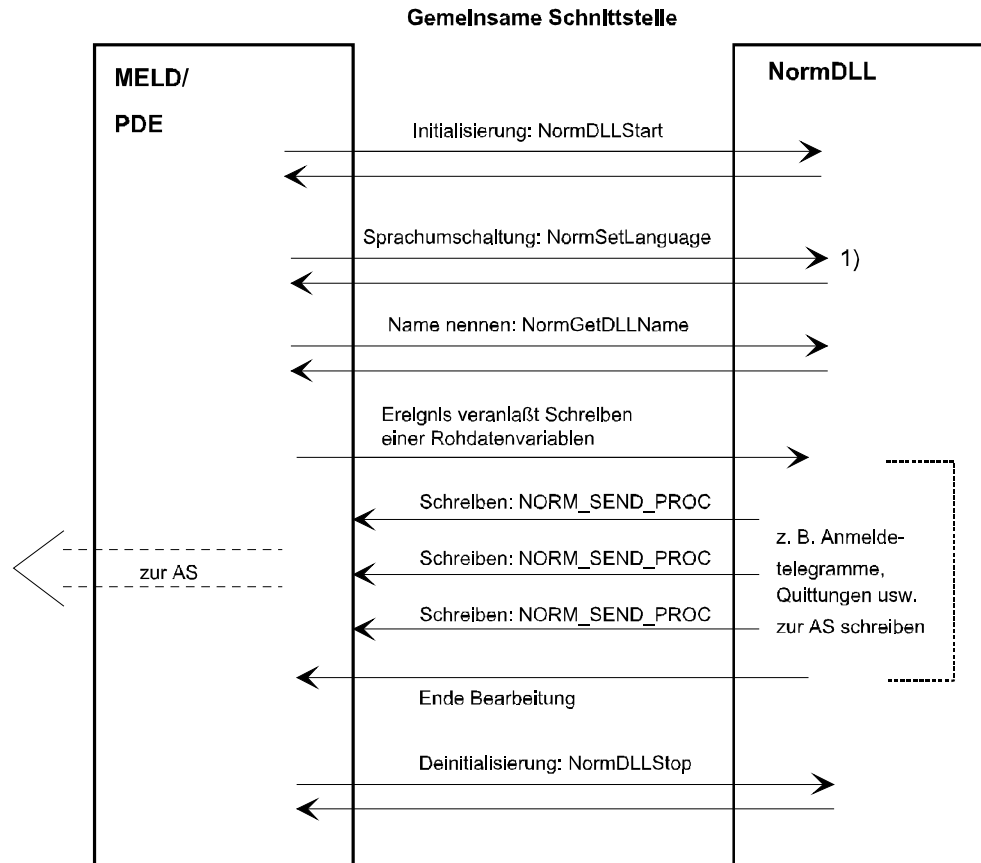
- OS zu AS: (Abgabe von An-/Abmeldeaufträgen, Quittierungen)
- AS zu OS: (Empfang von Meldungen und Archivdaten)

Mit einem Initialisierungsaufruf teilt TagLogging/AlarmLogging der S7PMC-DLL die projektierten Archivvariablenamen / Meldungsnummern mit. Für diese Objekte muß sich die Normierungs-DLL (WinCC) bei der AS anmelden. Der Initialisierungsaufruf kann zu jedem beliebigen Zeitpunkt bearbeitet werden,

Die Normierungs-DLL wird von AlarmLogging/TagLogging zur Deinitialisierung aufgerufen, um Ressourcen zurückzugeben usw.

5.3.1 Gemeinsame Schnittstelle zu AlarmLogging und TagLogging

Die allgemeinen Funktionen der Normierungs-DLL, die für AlarmLogging und TagLogging identisch sind, sind in einer gemeinsamen Schnittstelle zusammengefaßt. Die Funktionsnamen beginnen alle mit 'NORM...'.
(Präfix für AlarmLogging-spezifische Funktionen: 'Mld...', Präfix für TagLogging-spezifische Funktionen: 'Pde...'.)

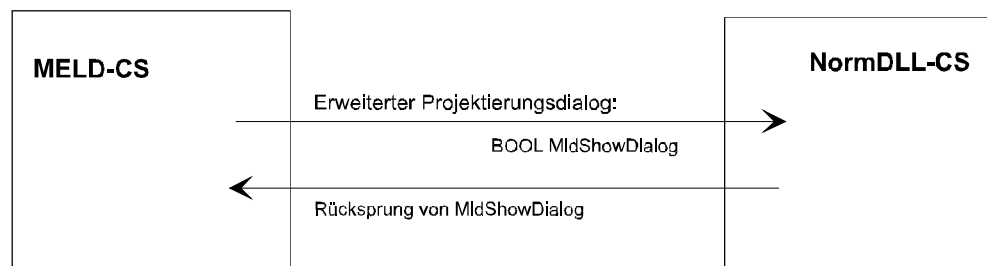
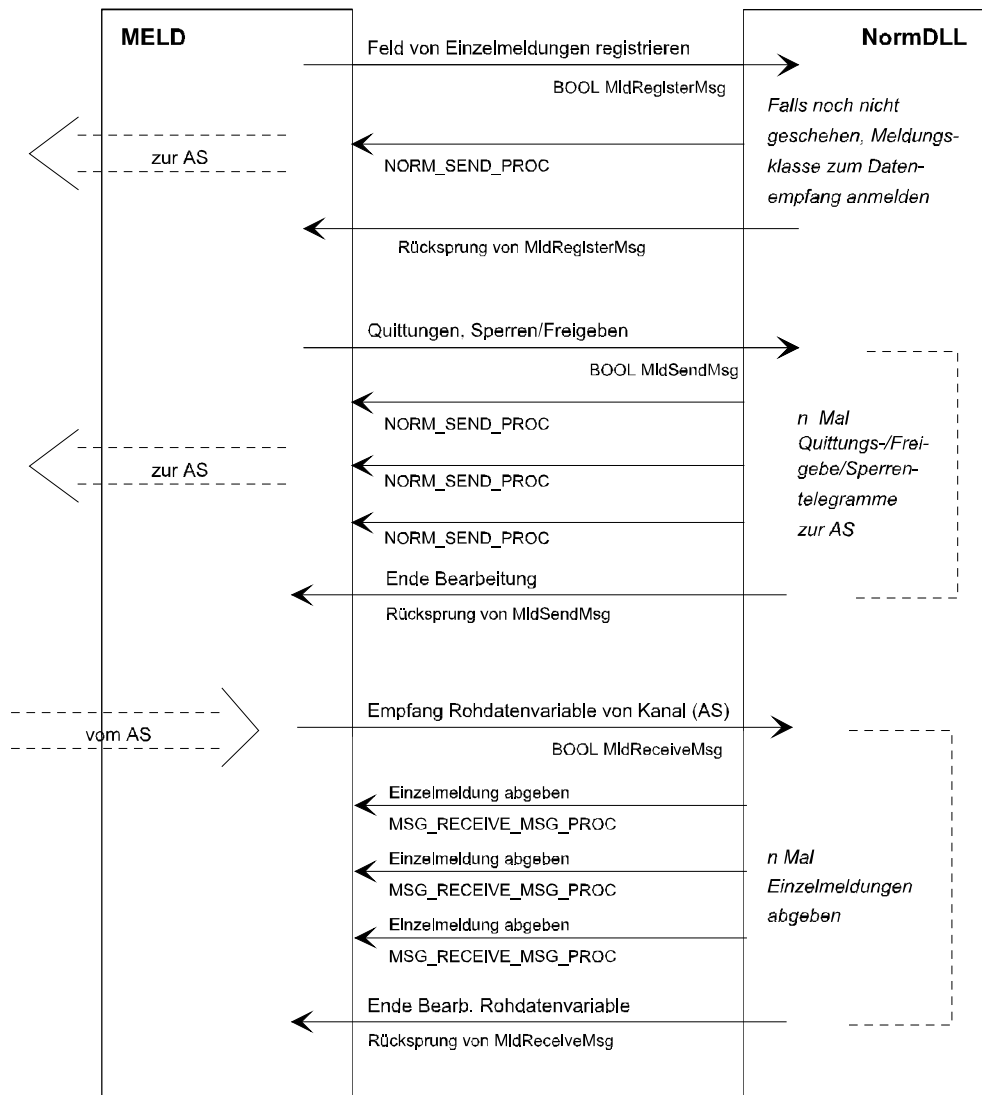


1) Die Sprachumschaltung ist nur bei Normierungs-DLL's erforderlich, die einen eigenen Dialog besitzen.

MELD = AlarmLogging

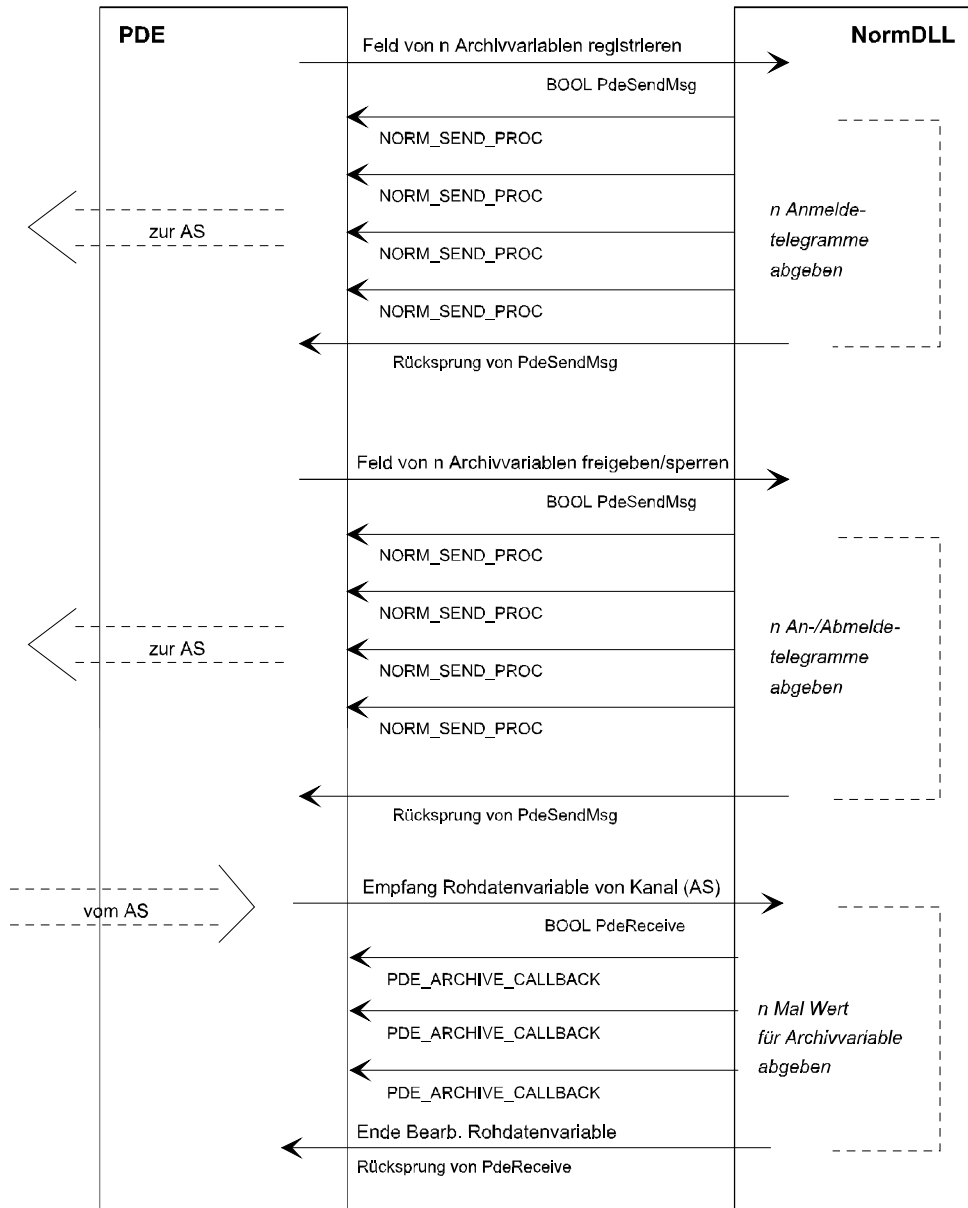
PDE = TagLogging

AlarmLogging-spezifische Zusätze
Runtime

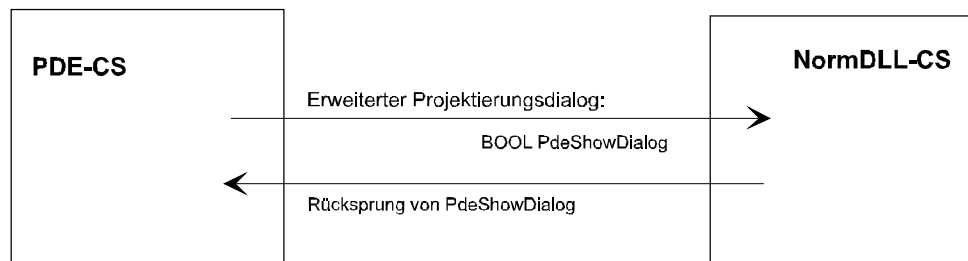


5.3.2 TagLogging-spezifische Zusätze

Runtime



Erweiterter Projektierungsdialog



5.3.3 API-Funktionen einer WinCC-Normierungs-DLL

Die Normierungs-DLL gliedert sich in folgende Teilbereiche:

- Initialisierung der Normierungs-DLL
 - Initialisierung durch das Betriebssystem während des Ladevorgangs der Normierungs-DLL (LibMain)
 - Abfrage der Eigenschaften einer Normierungs-DLL
 - Abfrage des Namens der Normierungs-DLL
- Shutdown der Normierungs-DLL
 - Shutdown durch TagLogging und AlarmLogging
 - Entladen durch das Betriebssystem
- Erweiterungen der Projektierung
 - Dialogerweiterung bei der Projektierung von Meldungen
 - Dialogerweiterung bei der Projektierung von Archivvariablen
- Online-Dienste
 - Registrierung aller Normierungs-DLL-spezifischen Objekte (Meldungen, Archivvariablen)
 - Sprachumschaltung
- Normierung
 - Normierung von Meldungen
 - Normierung von Archivvariablen

5.3.3.1 Initialisierung der Normierungs-DLL

Initialisierung während des Ladevorgangs

Die Applikationen AlarmLogging und/oder TagLogging laden eine WinCC-Normierungs-DLL mit Hilfe des Systemaufrufs *LoadLibrary*. Daraufhin wird die Normierungs-DLL durch das Betriebssystem geladen und durch dessen Standardmechanismen initialisiert. Alle Einsprungsadressen der Normierungs-DLL sind festgelegt.

5.3.3.2 Abfrage der Eigenschaften einer Normierungs-DLL

Mit dem Aufruf NormDLLStart meldet sich AlarmLogging und TagLogging bei der jeweiligen Normierungs-DLL an. Er ist zum Informationsaustausch zwischen der Normierungs-DLL und der Applikation gedacht.

NormDLLStart

```
include <winccnrm.h>

BOOL NormDLLStart(
    LPVOID lpUser,
    BOOL bModeRuntime,
    PNORM_STARTSTRUCT pcis,
    PCMN_ERROR lpError);
```

Parameter	Beschreibung
lpUser	Pointer auf Applikations-Daten, unverändert an Callback weitergeben
bModeRuntime	TRUE wenn Norm.-DLL im Runtime-Modus gestartet wird, FALSE wenn im Projektierungs-Modus; wird von Norm.-DLL derzeit nicht ausgewertet
pcis	Pointer auf Start-Struktur.
lpError	Pointer auf Standard-WinCC Fehlerstruktur

Return	Beschreibung
TRUE	Kein Fehler
FALSE	Fehler in API-Funktion, Beschreibung der Fehlerursache über den Zeiger lpError

NORM_STARTSTRUCT

Komponente	Beschreibung	E/A
dwSize	Größe der Struktur in Byte	A
lpstrProjectPath	Pfad des aktuell angewählten Projekts	E
NORM_SEND_PRO C pfnWriteRwData	Ptr auf Callback-Funktion der Applikation, über die die Normierungs-DLL eine Rohdatenvariable über den DM an das AS schickt.	E
dwAppID	Applikationskennung: 1 = AlarmLogging 2 = TagLogging 3 = USER (für weitere Applikat. reserviert, z.Z. nicht benötigt)	E
dwLocaleID	zum Zeitpunkt des Aufrufs aktuelle Spracheinstellung	E
dwNormCap	Eigenschaften der Normierungs-DLL nach folgender Tabelle	A

Die Callback-Funktion zum Schicken von Rohdatenvariablen an den WinCC-Datenmanager wird wie folgt versorgt:

```
typedef BOOL( *NORM_SEND_PROC ) (
    LPDPM_VAR_UPDATE_STRUCT    lpDmVarUpdate,
    DWORD                      dwWait,
    LPVOID                      lpUser,
    LPCMN_ERROR                 lpError );
```

Parameter	Beschreibung
lpDmVarUpdate	Zeiger auf Rohdatenvariable
dwWait	Kennung, ob die Applikation auf den Abschluß des Schreibaufrufs warten soll, oder nicht: WAIT_ID_NO mit SET_VALUE WAIT_ID_YES mit SET_VALUE_WAIT
lpUser	Pointer auf Applikations-Daten, beim Aufruf NormDLLStart gemerkt
lpError	Pointer auf Standard-WinCC Fehlerstruktur

Return	Beschreibung
TRUE	Kein Fehler
FALSE	Fehler in API-Funktion, Beschreibung der Fehlerursache über den Zeiger lpError

Jeder Eigenschaft ist ein Bit nach folgender Tabelle zugeordnet.

DEFINE	Bitmaske		Bedeutung
NORMCAP_DIALOG	0x00000001	gesetzt	Normierungs-DLL bietet spez. Dialog
		gelöscht	Normierungs-DLL bietet keinen Dialog
NORMCAP_REENTRANT	0x00000002	gesetzt	Normierungs-DLL ist reentrant-fähig
		gelöscht	Normierungs-DLL ist nicht reentrant-fähig
NORMCAP_MSG_FREE_LOCK	0x00000004	gesetzt	An-/Abmelden für Meldungen ist möglich.
		gelöscht	An-/Abmelden für Meldungen ist nicht möglich.
NORMCAP_ARC_FREE_LOCK	0x00000008	gesetzt	An-/Abmelden für Archivvariablen ist möglich.
		gelöscht	An-/Abmelden für Archivvariablen ist nicht möglich.
NORMCAP_MSG_GENERIC	0x00000010	gesetzt	Meldungen können generisch erzeugt werden.

DEFINE	Bitmaske		Bedeutung
		gelöscht	Meldungen können generisch nicht erzeugt werden.
NORMCAP_ARC_G ENERIC	0x00000020	gesetzt	Archivvariablen können generisch erzeugt werden.
		gelöscht	Archivvariablen können generisch nicht erzeugt werden.

5.3.3.3 Abfrage des Namens der Normierungs-DLL

NormGetDLLName

```
include <winccnrm.h>
LPTSTR NormGetDLLName( void );
```

Return	Beschreibung
LPTSTR	Zeiger auf einen String, der den Namen der Normierungs-DLL in Klartext enthält; der Name ist abhängig von der aktuellen Spracheinstellung.

5.3.4 Shutdown der Normierungs-DLL

Shutdown durch TagLogging und AlarmLogging

TagLogging und AlarmLogging verständigen die Normierungs-DLL, wenn die Applikationen beendet werden. In der Normierungs-DLL werden dann die Ressourcen ordnungsgemäß zurückgegeben.

NormDLLStop

```
include <winccnrm.h>

BOOL NormDLLStop (void);
```

Return	Beschreibung
TRUE	Funktion erfolgreich
FALSE	Fehler in API-Funktion

Entladen durch das Betriebssystem

Es sind keine besonderen Vorkehrungen erforderlich.

5.3.4.1 Erweiterungen der Projektierung

Für die S7PMC Objekte sind spezifische Angaben erforderlich. Diese Angaben werden zunächst in einem Dialog mit Standardmitteln (ohne MFC) erfragt und gehen direkt entweder in die WinCC-Meldungsnummer oder in den Namen der Archivvariablen ein. Damit muß die Normierungs-DLL diese Angaben nicht selbst ablegen und verwalten. Um die projektweite Eindeutigkeit einer Meldungsnummer oder Archivvariablen zu gewährleisten, ist eine Zuordnung zwischen Meldungsnummer bzw. Archivvariablen und der zugehörigen Rohdatenvariablen erforderlich. Diese Zuordnungsinformation ist Bestandteil der Meldungsnummer bzw. des Archivvariablennamens.

5.3.4.2 Dialogerweiterung bei der Projektierung von S7PMC-Meldungen

Die Normierungs-DLL hat eine API - Funktion zum Festlegen der S7PMC-spezifischen Meldungsnummer. Diese Funktion wird vom Meldesystem CS beim Parametrieren von Einzelmeldungen, die zu einer S7PMC-Normierungs-DLL gehören, aufgerufen. Als Meldungsnummer vergibt die S7PMC-Normierungs-DLL die Nummer, die aus zwei Teilen besteht:

Teil 1:

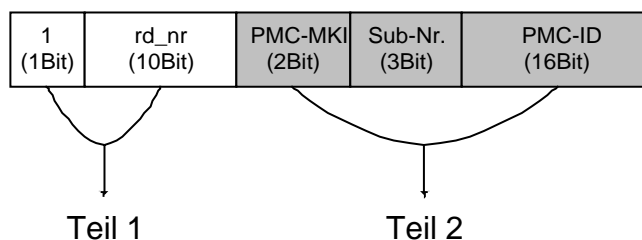
Nummer die projektweit eine AG-CPU eindeutig identifiziert (Rohdatenvariablen-Nummer)

Teil 2:

Nummer die AS-seitig zu der Meldung gehört und diese eindeutig innerhalb einer AG-CPU identifiziert (Normierungs-DLL-spezifisch)

Im Projektierungsdialog muß folgende Auswahl getroffen werden, um die Meldungsnummer aufzubauen:

Struktur eines S7PMC-Meldungsnummer (32 Bit)



zu Teil 1

Jede Meldung gehört zu einer Rohdatenvariablen, die eine AG-CPU identifiziert. Um die Zuordnung Rohdatenvariable – Meldungsnummer vornehmen zu können, wurde folgende Festlegung getroffen.

Der Name der Rohdatenvariable für S7PMC - und allen Verbindungsarten mit Normierungs-DLL - hat folgenden festen Aufbau:

@rd_alarm#rd_nr

@rd_alarm# fester Bestandteil des Namens einer Rohdatenvariablen für Normierungs-DLL's
 rd_nr Dezimalzahl von 0 - 1023 zur Identifikation einer Rohdatenvariablen (ohne führende Nullen)

Das höchstwertigste Bit der Meldungsnummer ist bei Meldungsnummern die von Normierungs-DLL's (extern) vergeben werden gesetzt. Diese Meldungen dürfen nur von den zugehörigen Normierungs-DLL's bearbeitet werden, d. h. über den Projektierungsdialog von AlarmLogging kann die Meldungsnummer nicht verändert werden.

Zu Teil 2

Dieser Teil der Meldungsnummer kann von der jeweiligen Normierungs-DLL belegt werden. Bei S7PMC hat er folgende Bedeutung:

MKI	Meldungsklasse; auszuwählen ist eine der Klassen: SCAN (1) ALARM/NOTIFY (2) ALARM_8P/ALARM_8 (2) LTM (3)
Sub-Nr	Submeldungsnummer nur relevant bei ALARM_8 und ALARM_8P: 1...8
PMC-ID	PMC-Meldungsnummer (Baustein-Eingangsparameter EV-ID): 1...16386 bei Meldungsklasse SCAN und ALARM/NOTIFY bzw. ALARM_8P/ALARM_8 1...7 bei Meldungsklasse LTM

MldShowDialog

```
include <winccnrm.h>

BOOL WINAPI MldShowDialog(
    HWND          hwnd,
    LPMSG_CSDATA_GENERIC lpMsgCS,
    LPDM_PROJECT_INFO lpDMProjectInfo,
    LPCMN_ERROR   lpError );
```

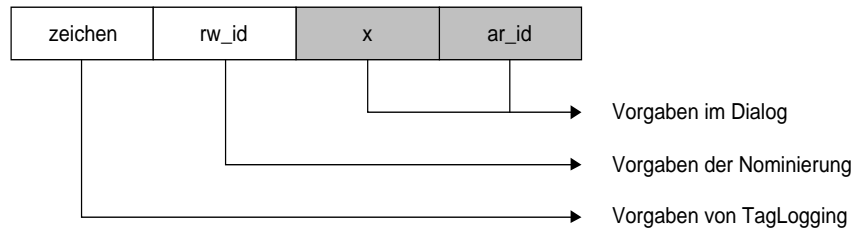
Parameter	Beschreibung
hwnd	Fensterhandle
lpMsgCS	Zeiger auf Einzelmeldungsdaten
lpDMProjectInfo	Zeiger auf ProjektInfostruktur
lpError	Zeiger auf Standard-WinCC Fehlerstruktur

Return	Beschreibung
TRUE	Funktion erfolgreich
FALSE	Fehler in API-Funktion, Beschreibung der Fehlerursache über den Zeiger lpError

5.3.4.3 Dialogerweiterung bei der Projektierung von Archivvariablen

Die Normierungs-DLL hat eine API - Funktion zum Festlegen der S7PMC-spezifischen Archivvariablenamen. Diese Funktion wird vom TagLogging CS beim Parametrieren von Archivvariablen, die zu einer S7PMC-Verbindung gehören, aufgerufen. Als Archivvariablenname vergibt die S7PMC-Normierungs-DLL einen Namen, der aus mehreren Komponenten besteht, u.a. enthält er die AS-seitig zu dem Archiv gehörende Nummer. Mit diesem Algorithmus sind die S7PMC-Archivnummern eindeutig in der WinCC-Archivvariablenbeschreibung enthalten, was im Runtimebetrieb zu einer schnellstmöglichen Zuordnung führt.

TagLogging garantiert, daß die Archivvariablenamen insgesamt eindeutig sind.

Struktur eines S7PMC-Archivvariablennamens (max. 18 Byte lang)

Bezeichnung	Länge in Byte	Vergeben von	Bedeutung
zeichen	9	TagLogging	Feste Zeichenfolge, die von TagLogging vergeben wird, besteht aus dem Namen der Normierungs-DLL und # als Trennzeichen. z. B. für S7PMC: NRMS7PMC, erscheint nicht an der Oberfläche
rw_id	8	TagLogging/ Norm.DLL	Rohdaten-ID in Hexa-Zeichen (inklusive führende Nullen), damit findet eine eindeutig Zuordnung zu der Rohdatenvariablen (Verbindung) statt, zu der die Archivnummer gehört. Der Namensanteil wird von der Norm.DLL anhand der TagLogging-Eingangsparameter gebildet.
x	1	Norm.DLL- CS-Anteil	S7PMC-spezifische Kennung zur Unterscheidung zwischen BSEND und AR_SEND: 'A' = AR_SEND 'B' = BSEND
ar_id	4	Norm.DLL- CS-Anteil	ID als Hexa-Zeichen (inklusive führende Nullen) Abhängig von Kennung x: S7PMC-spezifische Archivnummer AR_ID oder S7-spezifische R_ID beim BSEND

Beispiel eines für S7PMC erzeugten Archivvariablennamens: #00000001#A#0014

PdeShowDialog

```
include <winccnrm.h>

BOOL WINAPI PdeShowDialog(
    LPVOID          hwnd,
    LPTSTR          lpszArcVarName,
    DWORD           dwArcVarNameLength,
    LPDM_VARKEY     lpVarKey,
    LPCMN_ERROR     lpError
);
```


Parameter	Beschreibung
hwnd	Fensterhandle
lpzArcVarName	Zeiger auf Stringfeld zur Ablage des Normierungs-DLL-spez. Archivvariablen-Namenanteils
dwArcVarNameLength	max. Länge Normierungs-DLL-spez. Namenanteils
lpVarKey	Zeiger auf Varkey der Rohdatenvariablen
lpError	Zeiger auf WinCC-Fehlerstruktur

Return	Beschreibung
TRUE	Funktion erfolgreich
FALSE	Fehler in API-Funktion, Beschreibung der Fehlerursache über den Zeiger lpError

5.3.4.4 Online-Dienste

Alle Meldungen registrieren

Diese Funktion ist erforderlich, weil die Normierungs-DLL keine Projektierungsinformation über die relevanten Meldungen hat. Meldungen werden von dem AS aber erst geschickt, wenn sich die Applikation (WinCC) zum Empfang von Meldungen angemeldet hat. AlarmLogging ruft die Funktion MldRegisterMsg derzeit für jede relevante Meldung auf und übergibt damit der Normierungs-DLL Projektierungsinformation für die Einzelmeldung. Außer der Meldungsbeschreibung erhält die Normierungs-DLL einen Zeiger auf die Rohdatenvariable (Verbindung) der dieser Meldung zugeordnet ist. Damit kann die Normierungs-DLL zur Laufzeit im Hauptspeicher eine Tabelle anlegen, mit der die S7PMC-spezifischen Anmeldeprogramme aufgebaut werden können.

MldRegisterMsg

```
include <winccnrm.h>

BOOL WINAPI MldRegisterMsg(
    LPDM_VARKEY    lpDMVarKey,
    LPDWORD        lpMsgNumber,
    DWORD          DwNumMsgNumber,
    LPCMN_ERROR    lpError );
```

Parameter	Beschreibung
lpDMVarKey	Zeiger auf Varkey der Rohdatenvariablen
lpMsgNumber	Zeiger auf Feld mit Einzelmeldungsnummern
dwNumMsgNumber	Anzahl Einzelmeldungsnummern
lpError	Zeiger auf WinCC-Fehlerstruktur

Return	Beschreibung
TRUE	Funktion erfolgreich
FALSE	Fehler in API-Funktion, Beschreibung der Fehlerursache über den Zeiger lpError

5.3.4.5 Registrieren aller Archivvariablen

Diese Funktion ist erforderlich, weil die Normierungs-DLL keine Projektierungsinformation über die relevanten Archivvariablen hat. Deshalb wird die Funktion PdeSendMsg für eine bestimmte Anzahl relevanter Archivvariablen aufgerufen und damit die Projektierungsinformation und TagLogging-Zusatzinformation für die Archivvariablen bekanntgegeben.

Es können mehrere Archivvariablen einer Verbindung mit einem Aufruf registriert werden. Pro Archivvariablen übergibt TagLogging ein Doppelwort als Zusatzinformation an die Normierungs-DLL, die im Speicher der Normierungs-DLL gehalten wird. Diese Zusatzinformation wird von TagLogging benötigt, sobald Archivvariablen-Werte zu bearbeiten sind (in der Callback-Funktion TagLogging_ARCHIVE_CALLBACK). Damit kann die Normierungs-DLL zur Laufzeit im Hauptspeicher eine Tabelle anlegen, mit der die S7PMC-spezifischen Anmeldetelegramme für die jeweiligen Archive aufgebaut werden können. Die Anmeldetelegramme sind erforderlich, um dem AS die Empfangsbereitschaft für die jeweilige Archivnummer bekanntzugeben. Erst nach erfolgreichem Anmelden schickt das AS die Archivdaten an die Applikation (WinCC).

PdeSendMsg

```
include <winccnrm.h>

BOOL WINAPI PdeSendMsg (
    NORM_SEND_PROC      lpfnCallback,
    DWORD               dwFunctionId,
    LPSZ_ARC_VAR_NAME  lpszArcVarName,
    LPDWORD             lpdwData,
    DWORD               dwNumArchVarName,
    LPDM_VARKEY        lpVarKey,
    LPVOID              lpUser,
    LPCMN_ERROR         lpError
);
```

Parameter	Beschreibung
lpfnCallback	Zeiger auf Callback-Routine mit der die von der Normierungs-DLL aufgebaute Rohdatenvariable an den DM zu übergeben ist. Falls Null, wird die Callback-Routine aus der Ini-Struktur aufgerufen. Die Funktions-Adresse aus der Ini-Struktur ist nicht identisch mit diesem Parameter.
dwFunctionId	Funktionskennung FUNC_ID_REGISTER (vgl. folgende Tabelle), dieselbe Funktion gilt für alle aufgeführten Variablen
lpszArcVarName	Zeiger auf ein Pointerfeld, deren Elemente auf die Namen der Archivvariablen verweisen
lpdwData	Zeiger auf ein Feld dessen Elemente Zusatzdaten für die Archivvariablen enthalten, kann auch Null sein. Der einer Archivvariablen zugehörige Zusatzwert wird bei der Funktion FUNC_ID_REGISTER (Archivvariable anmelden) unverändert in interne

Parameter	Beschreibung
	Listen der Normierungs-DLL übernommen und zu gegebener Zeit an die TagLogging_ARCHIVE_CALLBACK weitergereicht. Bei den anderen Funktionskennungen ohne Bedeutung.
dwNumArchVarName	Anzahl Archivvariablenamen, die zu bearbeiten sind
lpVarKey	Zeiger auf Varkey der Rohdatenvariablen
lpUser	Zeiger auf Userdaten, unverändert an Callback übergeben
lpError	Zeiger auf WinCC-Fehlerstruktur

Return	Beschreibung
TRUE	Funktion erfolgreich
FALSE	Fehler in API-Funktion, Beschreibung der Fehlerursache über den Zeiger lpError

Mögliche Funktionen der Prozedur PdeSendMsg (Werte von dwFunctionId):

DEFINE	Bitmaske	Bedeutung
FUNC_ID_LOCK	0x00000001	Archivvariable sperren
FUNC_ID_FREE	0x00000002	Archivvariable freigeben
FUNC_ID_REGISTER	0x00000004	Archivvariable anmelden
FUNC_ID_UNREGISTER	0x00000008	Archivvariable abmelden (wird derzeit nicht benötigt)

5.3.4.6 Sprachumschaltung

Der Projektierungsdialog muß sprachabhängig sein, d. h. die Normierungs-DLL muß die aktuell eingestellte Sprache kennen. Beim Start wird die Spracheinstellung in der Startstruktur mitgegeben. Die dynamische Sprachumschaltung muß von TagLogging und AlarmLogging auch an die Normierungs-DLL weitergereicht werden. Dafür gibt es den Aufruf

NormSetLanguage

```
include <winccnrm.h>

BOOL NormSetLanguage(
    DWORD          dwLocaleID,
    LPCMN_ERROR    lpError
);
```

Parameter	Beschreibung
dwLocaleID	zum Zeitpunkt des Aufrufs aktuelle Spracheinstellung
lpError	Zeiger auf Standard-WinCC Fehlerstruktur

Return	Beschreibung
TRUE	Funktion erfolgreich
FALSE	Fehler in API-Funktion, Beschreibung der Fehlerursache über den Zeiger lpError

5.3.5 Normierung

Hat sich eine Applikation für den Empfang von Meldungen oder Archivdaten beim AS angemeldet, erhält sie diese Daten über die jeweilige Rohdatenvariable. Das Anmelden führt die Normierungs-DLL beim Registrieren durch. Ab diesem Zeitpunkt können Datentelegramme vom AS kommen. Die Datentelegramme werden in Rohdatenvariablen verpackt und über die Kanal-DLL, den Datenmanager und die jeweilige Applikation (hier TagLogging oder AlarmLogging) an die Normierungs-DLL weitergereicht, die für den Typ der Rohdatenvariablen zuständig ist. Die Normierungs-DLL interpretiert die ankommenden Daten und bildet daraus Meldungen bzw. Archivdaten.

5.3.5.1 Ableitung von Einzelmeldungen

In dem Inhalt einer Rohdatenvariablen (eines Telegramms) können n Einzelmeldungen abgelegt sein. Die Normierungs-DLL muß dieses S7PMC-spezifisches Telegramm interpretieren und die sich daraus ergebenden Einzelmeldungen an AlarmLogging weiterreichen.

Die Meldungsnummer (EV_ID) von S7PMC ist ein Teil der WinCC-Meldungsnummer. Mit einer Meldung können von S7PMC bis zu max. 10 Prozeßwerte mitgeliefert werden. Dabei ist als Prozeßwert auch der Typ „string“ erlaubt. Diesen Prozeßwerttyp unterstützt AlarmLogging nicht, derartige Zusatzwerte muß die Normierungs-DLL verwerfen.

Die Funktion MldReceiveMsg wird von AlarmLogging auch jedes Mal aufgerufen, wenn sich der Status der Rohdatenvariablen geändert hat, also wenn Status gestört nach OK oder umgekehrt vom DM festgestellt wird. Die Status-(Zustands-)Änderung der Rohdatenvariablen (entspricht einer Verbindung) ist für die S7PMC-Normierungs-DLL von Bedeutung. Näheres dazu ist im Kapitel „Bearbeitung beim Zustandswechsel“ beschrieben.

MldReceiveMsg

```
include <winccnrm.h>

BOOL WINAPI MldReceiveMsg(
    MSG_RECEIVE_MSG_PROC    lpfnMsgReceive,
    LPDM_VAR_UPDATE_STRUCT  lpDMVar,
    LPVOID                  lpUser,
    LPCMN_ERROR              lpError );
```

Parameter	Beschreibung
lpfnMsgReceive	Zeiger auf Callback-Routine mit der die von der Normierungs-DLL aufgebaute Einzelmeldung an AlarmLogging zu übergeben ist.
lpDMVar	Zeiger auf Rohdatenvariable
lpUser	Zeiger auf Userdaten, unverändert an Callback übergeben
lpError	Zeiger auf WinCC-Fehlerstruktur

Return	Beschreibung
TRUE	Funktion erfolgreich
FALSE	Fehler in API-Funktion, Beschreibung der Fehlerursache über den Zeiger lpError

Die Callback-Funktion zum Abgeben der Einzelmeldungen an AlarmLogging wird wie folgt versorgt:

```
typedef BOOL( *MSG_RECEIVE_MSG_PROC )(
    LPMSG_RTCREATE_STRUCT    lpMsgCreate ,
    DWORD                    dwNumMsg ,
    LPVOID                    lpUser ,
    LPCMN_ERROR               lpError );
```

Parameter	Beschreibung
lpMsgCreate	Zeiger auf eine WinCC - Meldung
dwNumMsg	Anzahl Einzelmeldungen
lpUser	Pointer auf Applikations-Daten
lpError	Pointer auf Standard-WinCC Fehlerstruktur

Return	Beschreibung
TRUE	Funktion erfolgreich
FALSE	Fehler in API-Funktion, Beschreibung der Fehlerursache über den Zeiger lpError

5.3.5.2 Meldungen quittieren, sperren/freigeben

Das Melde- und Quittierkonzept von WinCC-AlarmLogging und S7PMC sieht vor, daß Meldungen, abhängig von ihrer Projektierung, quittiert werden können. Die Quittungsinformation ist AlarmLogging bekannt, muß aber auch im Meldequittierspeicher des AS geführt werden. Um das zu erreichen, schickt AlarmLogging Quittungstelegramme über die der Verbindung entsprechenden Normierungs-DLL an das AS.

Die S7PMC-Normierungs-DLL baut anhand dieser Eingangsdaten die zugehörigen S7PMC-Telegramme auf, die über die AlarmLogging-Callback-Funktion NORM_SEND_PROC an den DM weitergereicht werden.

Dasselbe Verfahren gilt wenn eine Einzelmeldung von AlarmLogging gesperrt / wieder freigegeben - d. h. ihre Erzeugung an der Quelle im AS wird unterbunden / wieder freigeschaltet - werden soll.

MldSendMsg

```
include <winccnrm.h>

BOOL WINAPI MldSendMsg(
    NORM_SEND_PROC    lpfnMsgSend,
    LPMSG_SEND_DATA_STRUCT lpSendData,
    DWORD              dwNumData,
    LPVOID              lpUser,
    LPCMN_ERROR        lpError );
```

Parameter	Beschreibung
lpfnMsgSend	Zeiger auf AlarmLogging-Callback-Routine, mit der die von der Normierungs-DLL aufgebaute Rohdatenvariable zum Schreiben an das AS zu übergeben ist. Die Parameter sind in Kap. „Abfrage der Eigenschaften einer Normierungs-DLL“ beschrieben
lpSendData	Zeiger auf Sendedaten, Aufbau nachfolgend beschrieben

Parameter	Beschreibung
dwNumData	Anzahl zu bearbeitender Einzelaufträge
lpUser	Zeiger auf Userdaten, unverändert an Callback übergeben
lpError	Zeiger auf WinCC-Fehlerstruktur

Return	Beschreibung
TRUE	Funktion erfolgreich
FALSE	Fehler in API-Funktion, Beschreibung der Fehlerursache über den Zeiger lpError

Aufbau der AlarmLogging-Sendedaten (Einzelauftrag)

Variable	Bedeutung
DWORD dwVarID	Rohdatenvariablen - ID des DM
DWORD dwNotify	Notify Mögliche Werte MSG_STATE_QUIT Meldung quittieren MSG_STATE_LOCK Meldung sperren MSG_STATE_UNLOCK Meldung freigeben MSG_STATE_QUIT_EMERGENCY Alle Meldungen quittieren
DWORD dwData	bei QUIT, LOCK, UNLOCK --> Meldungsnummer bei NOTQUIT --> unbenutzt

5.3.5.3 Bearbeitung beim Zustandswechsel

Der Statuswechsel einer Verbindung (Rohdatenvariablen) muß der Normierungs-DLL bekannt gegeben werden. Dies geschieht über die Funktion **MldReceiveMsg**.

Statuswechsel von - nach	Bearbeitung in der S7PMC-Normierungs-DLL
gestört - OK	Anmeldetelegramme für alle S7PMC-Meldungsklassen, für die mindestens eine Meldung projiziert wurde, an das AS abgeben. Die Anmeldung wird S7PMC-Meldungsklassen-spezifisch vorgenommen. Die Normierungs-DLL kennt schon alle projizierten Meldungen aufgrund derer Registrierung.
OK - gestört	Die Normierungs-DLL muß aktive Aufträge verwerfen, die schon an das AS geschickt wurden, aber wegen des Zustandswechsels nicht mehr vollständig bearbeitet werden können (Quittungen fehlen).

5.3.5.4 Meldungsupdate der S7PMC-Normierungs-DLL

Beim Meldungsupdate liest die S7PMC-Normierungs-DLL den Meldezustand aller ihr durch das Registrieren bekanntgegebenen Meldungen und schickt ihn als Einzelmeldung an AlarmLogging. Somit kann beim Systemanlauf auf ein konsistentes Meldebild aufgesetzt werden.

Ein Meldungsupdate ist erforderlich, wenn

- ein Zustandswechsel von gestört nach OK erkannt wurde (das ist implizit auch beim Systemanlauf der Fall)
- das AS ein „Meldungsupdate-Telegramm“ schickt. Dieses Telegramm wird an jeden angemeldeten Teilnehmer geschickt, wenn z. B. ein Meldungsüberlauf festgestellt wird, wenn Meldungen von anderen Teilnehmern quittiert oder freigegeben werden.

Beim Meldungsupdate schickt das AS die Melde-Quittierzustände und die Sperrkennungen. Die Zusatzwerte und die Uhrzeit der Meldungen werden nicht geschickt. Die Normierungs-DLL versorgt in diesem Fall die Uhrzeit der Einzelmeldung mit der aktuellen Systemzeit und odert in den Meldestatus die Kennung MSG_STATE_UPDATE.

5.3.5.5 Normierung von Archivvariablen

Für TagLogging stellt die Normierungs-DLL zwei Funktionen zur Verfügung:

- Ableitung von einzelnen Archivvariablen-Werten aus dem Inhalt einer Rohdatenvariablen
- Sperren / Freigeben von Archivvariablen

5.3.5.6 Ableitung von einzelnen Archivvariablen-Werten

In dem Inhalt einer Rohdatenvariablen (eines Telegramms) können n Archivvariablen-Werte abgelegt sein. Die Normierungs-DLL muß dieses S7PMC-spezifisches Telegramm interpretieren und die sich daraus ergebenden Archivvariablen-Werte an TagLogging weiterreichen.

Für eine Archivvariable können Prozeßwertumformer mitgeschickt werden. Die S7PMC-Normierungs-DLL nimmt dann die gewünschte Umrechnung von Prozeßwert in Archivvariablen-Wert vor. Dabei werden in WinCC schon existierende Skalierungsfunktionen eingesetzt. Die genaue Vorgehensweise muß noch festgelegt werden.

Die Funktion PdeReceive wird von TagLogging auch jedes Mal aufgerufen, wenn sich der Status der Rohdatenvariablen geändert hat, also wenn Status gestört nach OK oder umgekehrt vom DM festgestellt wird. Die Status-(Zustands-)Änderung der Rohdatenvariablen (entspricht einer Verbindung) ist für die S7PMC-Normierungs-DLL von Bedeutung. Näheres dazu ist im Kapitel „Bearbeitung beim Zustandswechsel“ beschrieben.

PdeReceive

```
include <winccnrm.h>

BOOL PdeReceive (
    LPDmVarUpdateStruct    lpDmVarUpdate,
    TagLogging_ARCHIVE_CALLBACK lpfnCallBack,
    LPVOID                 lpUser,
    LPCMN_ERROR            lpError
);
```

Parameter	Beschreibung
lpDmVarUpdate	Zeiger auf Rohdatenvariable
lpfnCallBack	Zeiger auf Callback-Routine, mit der die Normierungs-DLL die einzelnen Archivvariablen-Werte an TagLogging übergibt.
lpUser	Zeiger auf Userdaten, unverändert an Callback übergeben
lpError	Zeiger auf WinCC-Fehlerstruktur

Return	Beschreibung
TRUE	Funktion erfolgreich
FALSE	Fehler in API-Funktion, Beschreibung der Fehlerursache über den Zeiger lpError

Die Callback-Funktion zum Abgeben der einzelnen Archivvariablen-Werte an TagLogging wird wie folgt versorgt:

```
OOB (*PDE_ARCHIVE_CALLBACK) (
    LPTSTR                lpszArcVarName,
    double                doValue,
    SYSTEMTIME*           lpstTime,
    DWORD                 dwFlags,
    DWORD                 dwData,
    LPVOID                lpUser,
    LPCMN_ERROR           lpError
);
```

Parameter	Beschreibung
lpszArcVarName	Archivvariablen-Name ab Rohdaten-ID
doValue	Archivvariablen-Wert
lpstTime	Zeiger auf Zeitstempel, aus den Nutzdaten der Rohdatenvariablen abgeleitet
dwFlags	Kennungen, deren genaue Bedeutung noch festgelegt wird.
dwData	Zusatzdatum, das beim Registrieren der Archivvariablen mitgegeben wurde, unverändert übergeben
lpUser	Zeiger auf Userdaten, unverändert von Funktionsaufruf übernommen
lpError	Zeiger auf WinCC-Fehlerstruktur

Return	Beschreibung
TRUE	Funktion erfolgreich
FALSE	Fehler in API-Funktion, Beschreibung der Fehlerursache über den Zeiger lpError

5.3.5.7 Sperren / Freigeben von Archivvariablen

Mit dieser Funktion nutzt TagLogging die Möglichkeit in S7PMC, den Empfang von Archivvariablenwerten zu steuern. Die S7PMC-Normierungs-DLL bildet dann einen Aufruf zum Abmelden bzw. wieder Anmelden des jeweiligen Archivs und gibt diesen Aufruf über die NORM_SEND_PROC an den DM weiter.

Das Sperren / Freigeben von Archivvariablen ist aus der Sicht der S7PMC-Normierungs-DLL fast identisch mit den Funktionen, die beim Registrieren einer Archivvariablen erforderlich sind. Für beide Funktionen wird deshalb dieselbe Funktion in der Normierungs-DLL aufgerufen PdeSendMsg.

Über die Funktionskennung dwFunctionId wird zwischen Registrieren und Sperren / Freigeben unterschieden: beim Sperren / Freigeben haben die Zusatzdaten pro Archivvariable lpdwData keine Bedeutung. Vgl. Kap. „Registrieren aller Archivvariablen“.

5.3.5.8 Bearbeitung beim Zustandswechsel

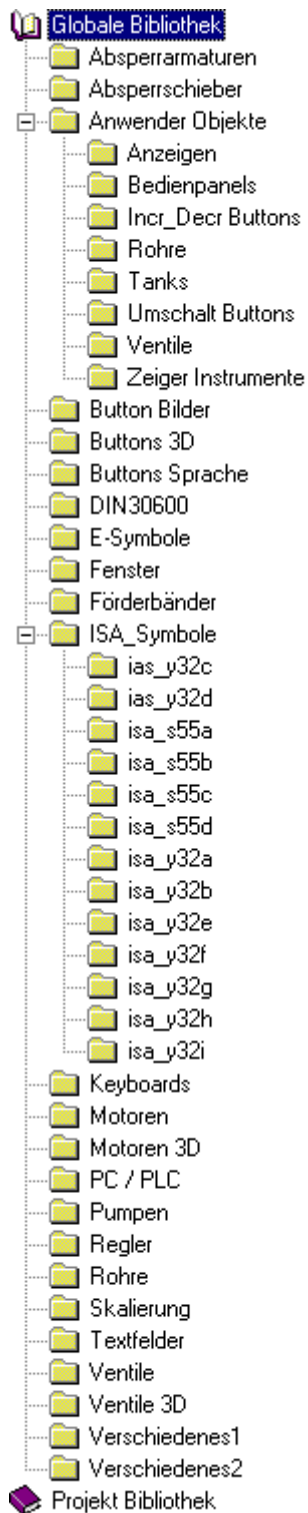
Der Statuswechsel einer Verbindung (Rohdatenvariablen) muß der Normierungs-DLL bekannt gegeben werden, dies geschieht über die Funktion **PdeReceive**.

Statuswechsel von - nach	Bearbeitung in der S7PMC-Normierungs-DLL
gestört - OK	Anmeldetelegramme für alle Archivvariablen aller Verbindungen Die Normierungs-DLL kennt schon alle projektierten Archivvariablen aufgrund derer Registrierung.
OK - gestört	Die Normierungs-DLL muß aktive Aufträge verwerfen, die schon an das AS geschickt wurden aber wegen des Zustandswechsels nicht mehr vollständig bearbeitet werden können (Quittungen fehlen).

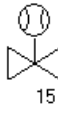
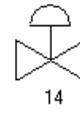
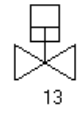
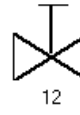
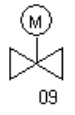
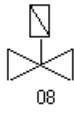
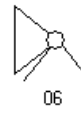
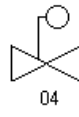
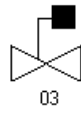
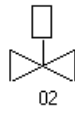
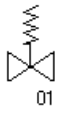
5.4 Erstellung von Bildbausteinen für WinCC

Dieses Kapitel gibt es nicht mehr. Es wird ersetzt durch die Dokumentation zu der WinCC Option: *IndustrialX*.

5.5 Globale Bibliothek



5.5.1 Absperrarmaturen



5.5.2 Absperrschieber



01



02



03



04



05



06



07



08



09



10



11



12



13



14



15



16



17



18



19



20



21



22

5.5.3 Anwender Objekte

5.5.3.1 Anzeigen



8-Bit Anzeige



8-Bit Anzeige +
I/O Field



Digitalausgabe

5.5.3.2 Bedienpanels



1_Slider



2_Slider



4_Slider

5.5.3.3 Incr_Decr_Buttons



Decrement_-1



Decrement_-Step



Increment_+1



Increment_+Step

5.5.3.4 Rohre



3D Rohr
horizontal



3D Rohr vertikal

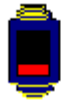


3D Rohr Winkel

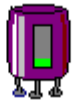
5.5.3.5 Tanks



Tank1



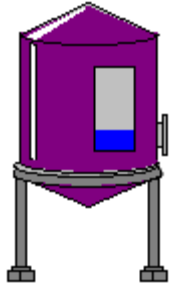
Tank2



Tank3



Tank4



5.5.3.6 Umschalt Buttons



Ein_Aus_1



Ein_Aus_2



Ein_Aus_3



Ein_Aus_4



Ein_Aus_5



Ein_Aus_6



Ein_Aus_7

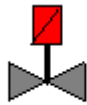


Ein_Aus_8

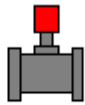
5.5.3.7 Ventile



Ventil001



Ventil002



Ventil003

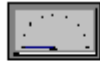


Ventil004

5.5.3.8 Zeiger Instrumente



Meter1_0-100



Meter1_Min-Max

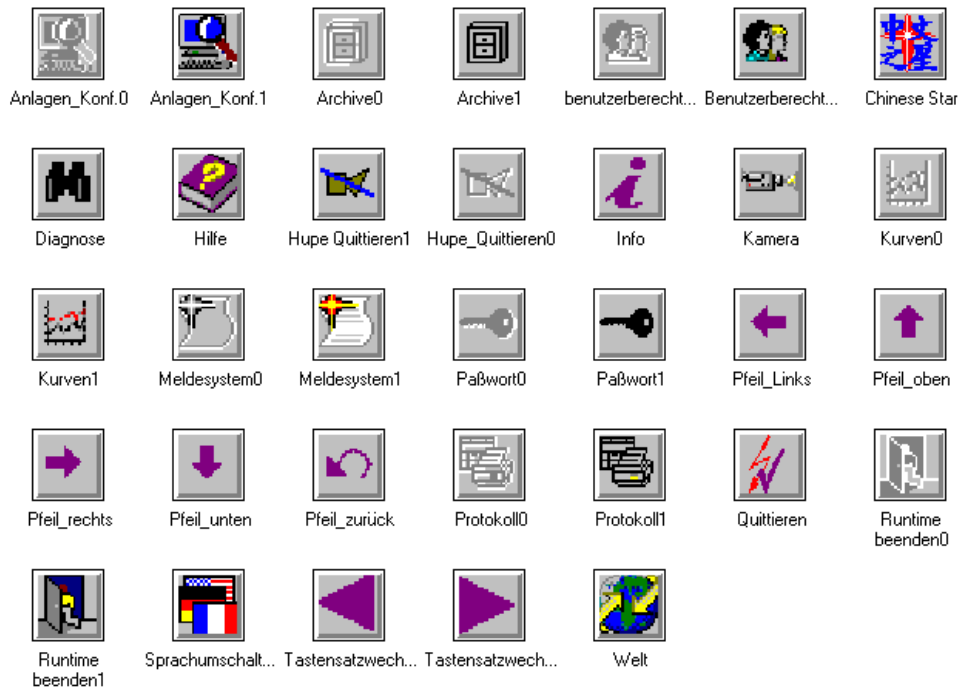


Meter2_Min-Max

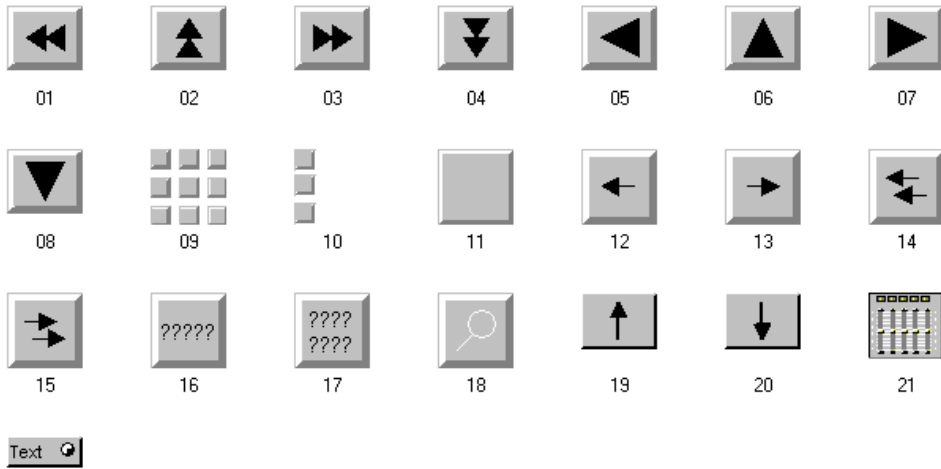


Meter3_Min-Max

5.5.4 Button Bilder

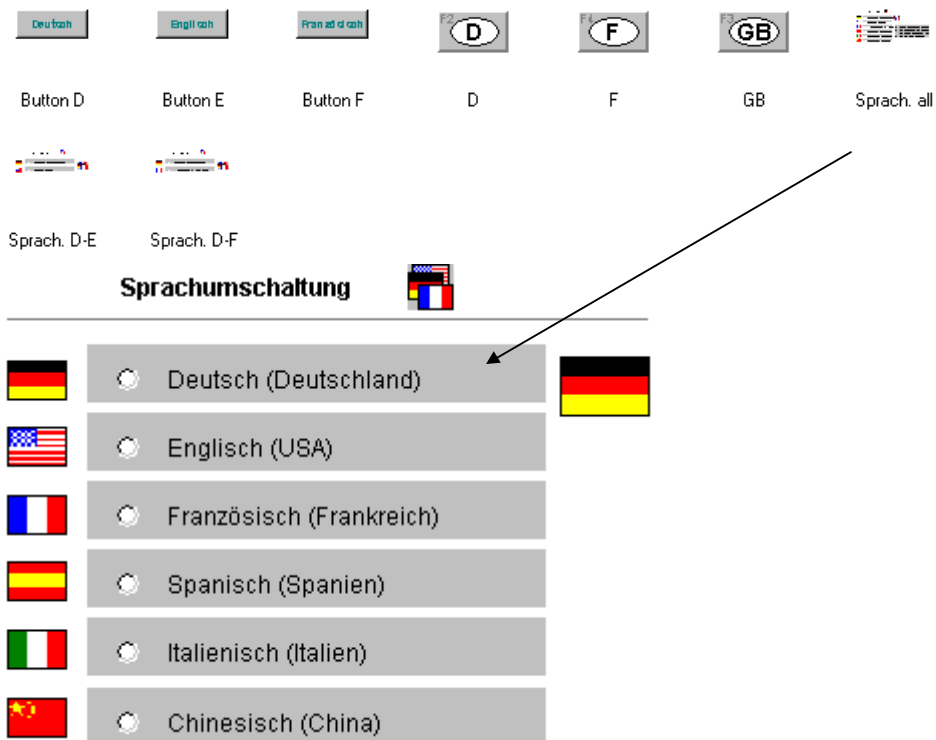


5.5.5 Buttons 3D

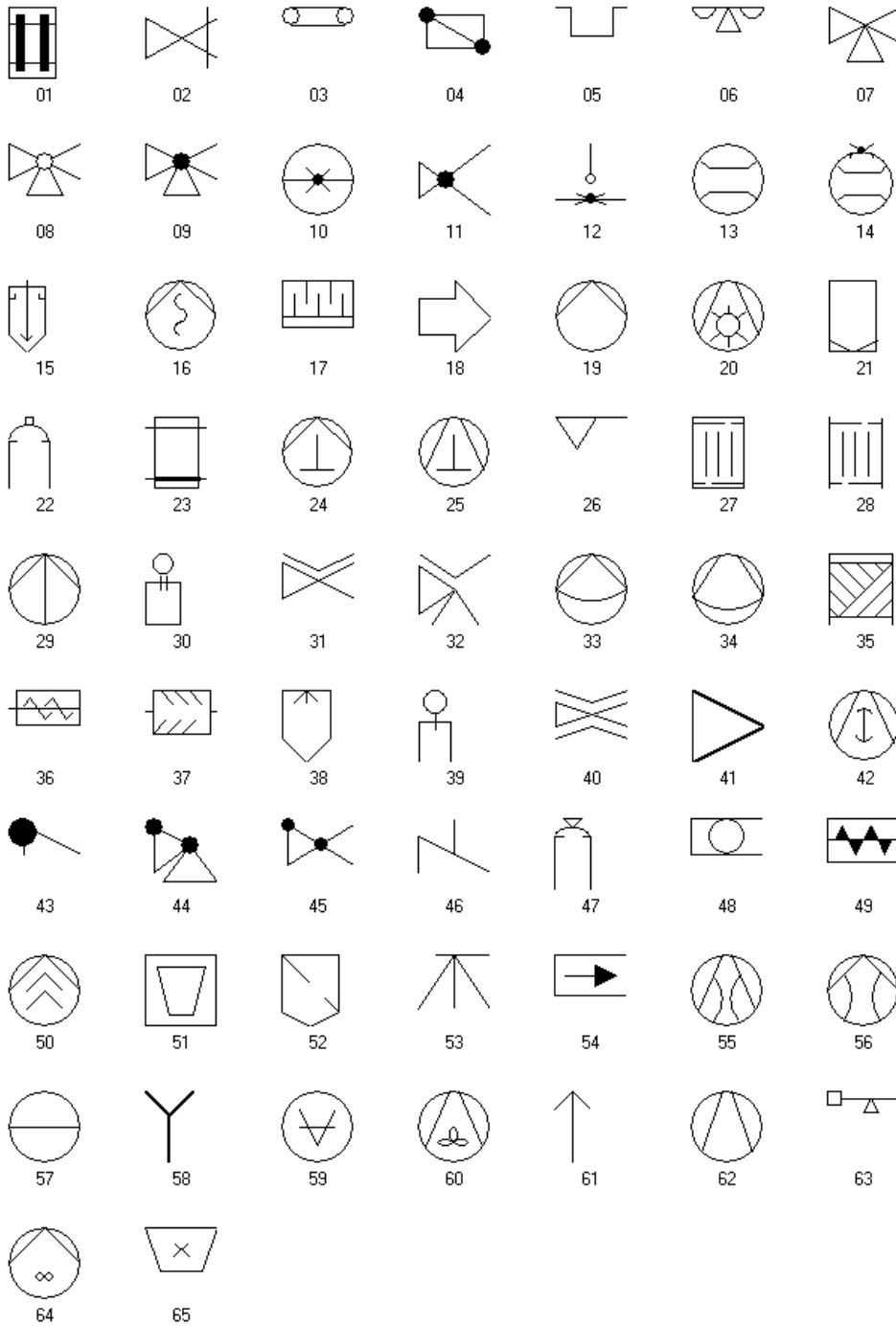


22

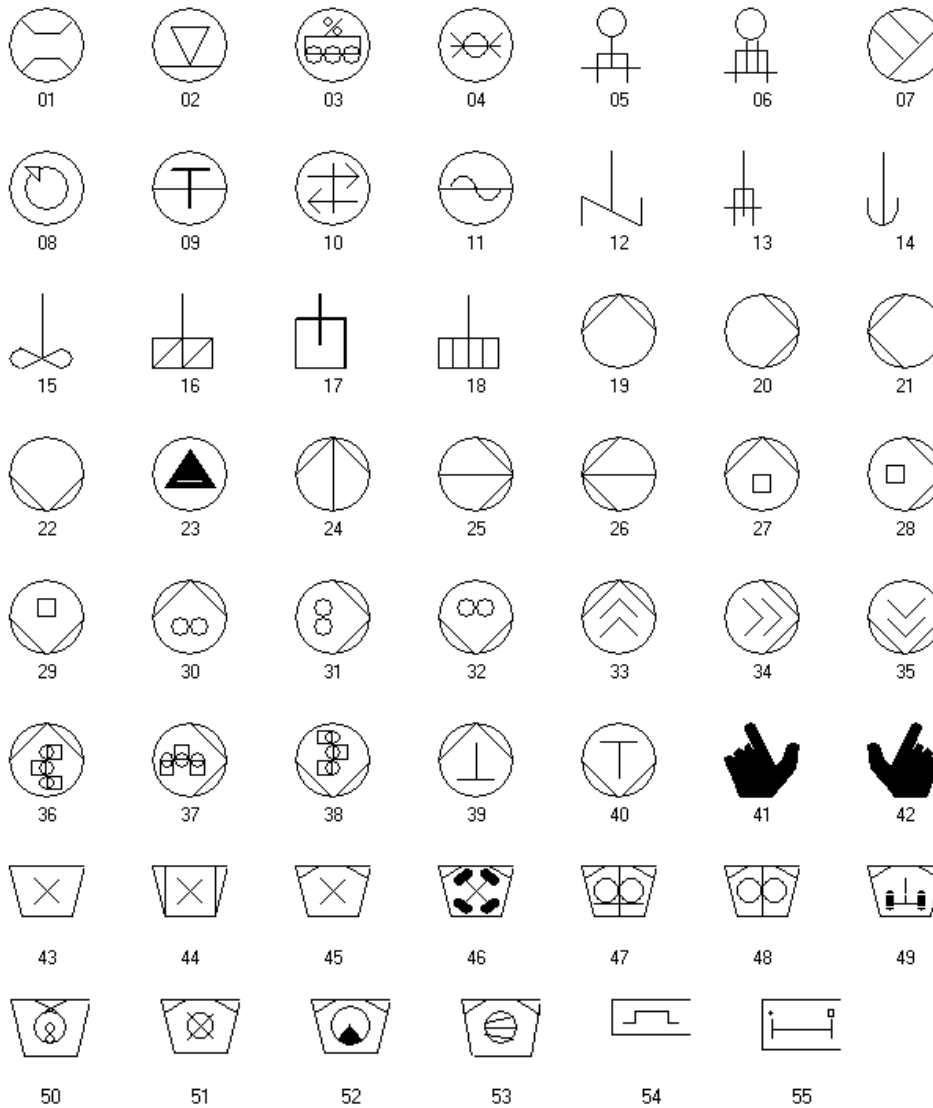
5.5.6 Buttons Sprachumschaltung



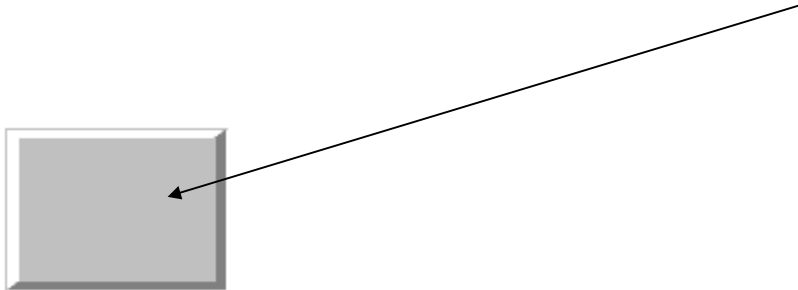
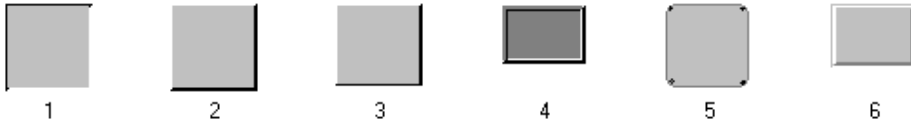
5.5.7 DIN30600



5.5.8 E-Symbole



5.5.9 Fenster



5.5.10 Förderbänder



1



2



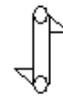
3



4



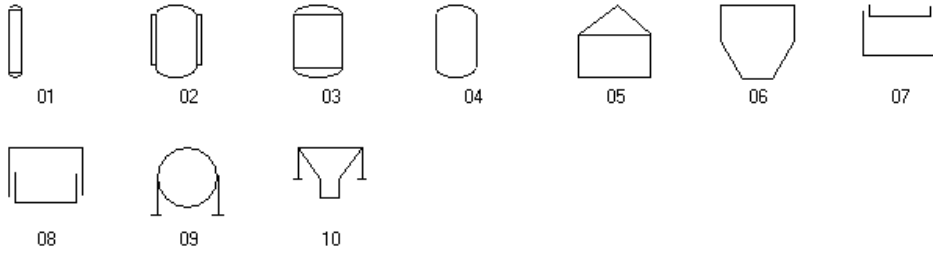
5



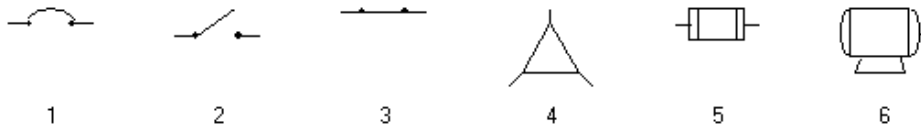
6

5.5.11 ISA Symbole

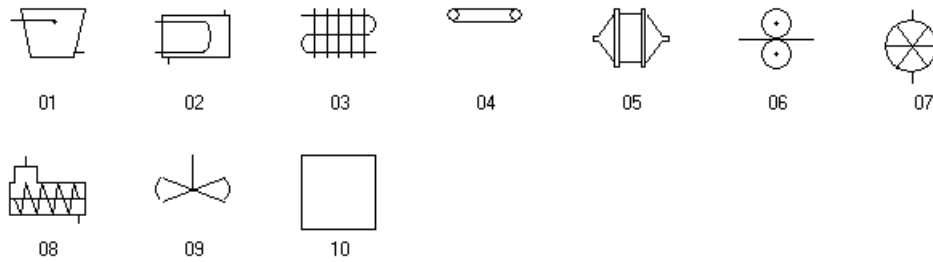
5.5.11.1 isa_s55a



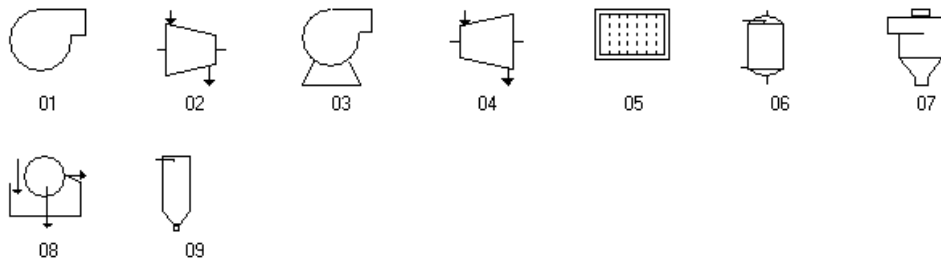
5.5.11.2 isa_s55b



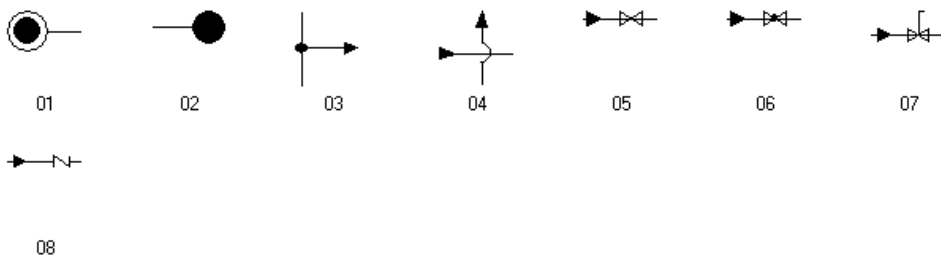
5.5.11.3 isa_s55c



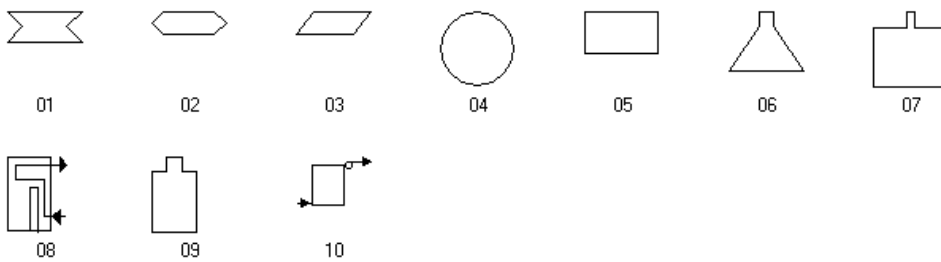
5.5.11.4 isa_s55d



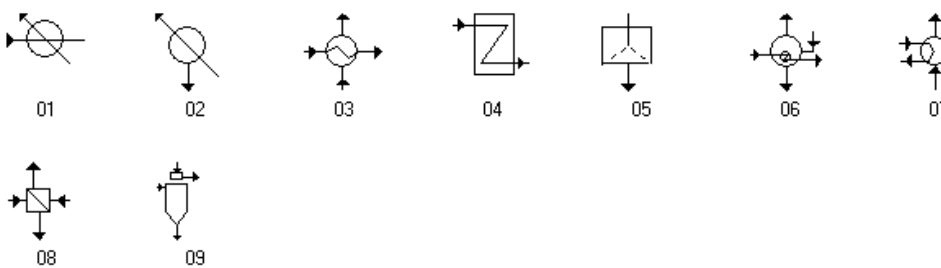
5.5.11.5 isa_y32a



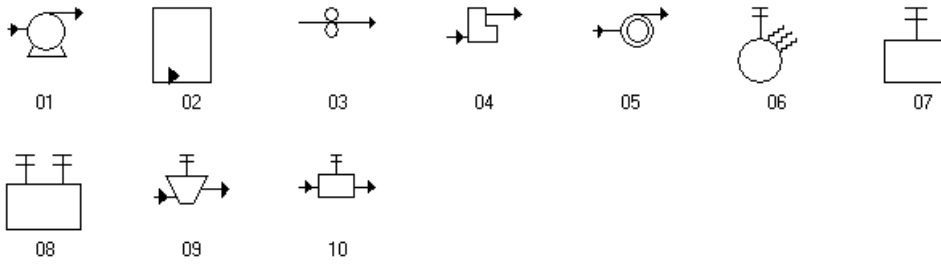
5.5.11.6 isa_y32b



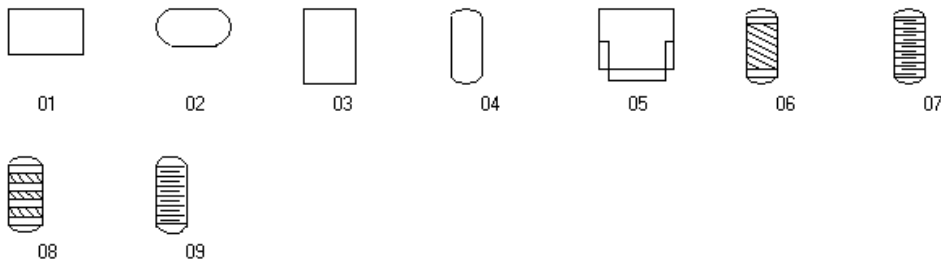
5.5.11.7 isa_y32c



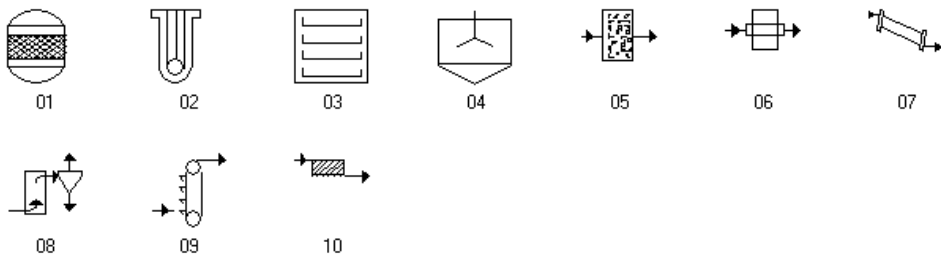
5.5.11.8 isa_y32d



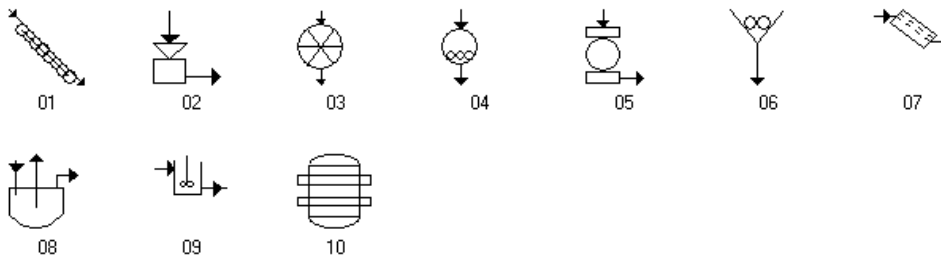
5.5.11.9 isa_y32e

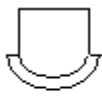


5.5.11.10 isa_y32f



5.5.11.11 isa_y32g

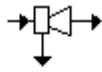


5.5.11.12 isa_y32h

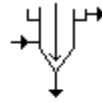
01



02



03



04

5.5.11.13 isa_y32i

01



02



03



04



05



06



07

5.5.12 Keyboards



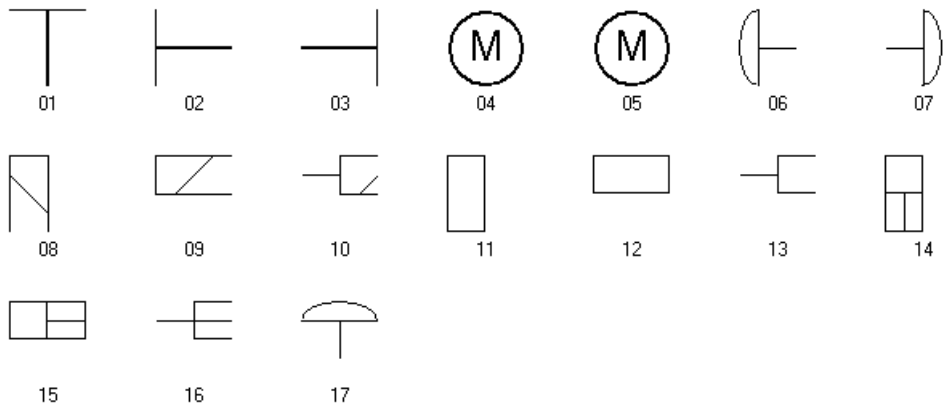
Keyboard

Keyboard char

Keyboard Nr.



5.5.13 Motoren



5.5.14 Motoren 3D



B3



B5



Motor001



Motor002



Motor003



Motor004



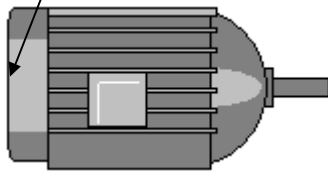
Motor005



Motor006



Motor007



5.5.15 PC / PLC



PC



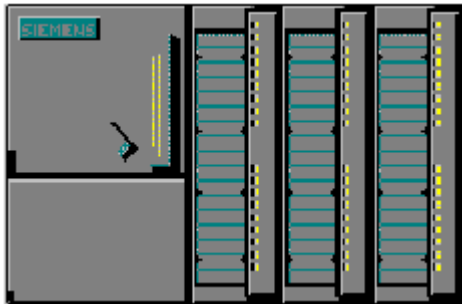
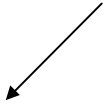
PLC S7 400



Screen WinCC1



Screen WinCC2



5.5.16 Pumpen



Pumpe001



Pumpe002



Pumpe003



Pumpe004



Pumpe005



Pumpe006



Pumpe007



Pumpe008



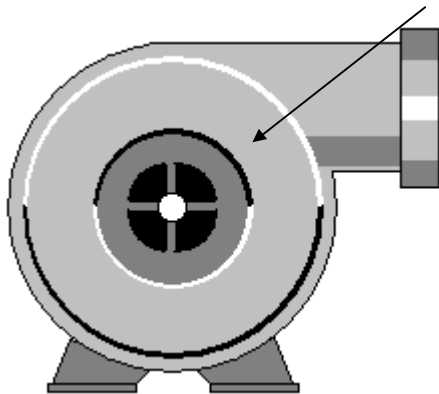
Pumpe009



Pumpe010



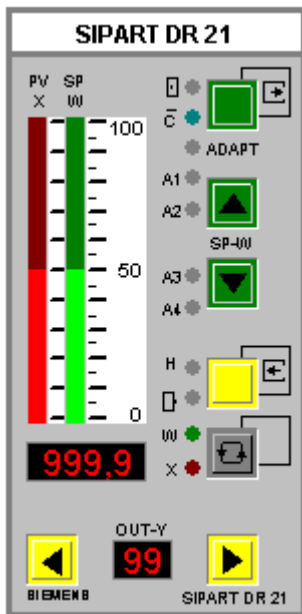
Pumpe011



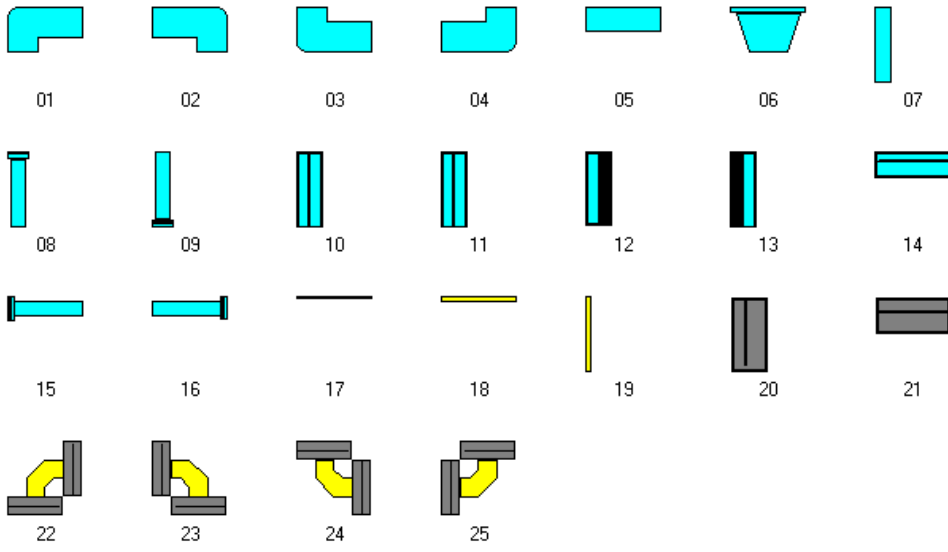
5.5.17 Regler



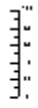
DR 21



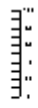
5.5.18 Rohre



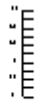
5.5.19 Skalierung



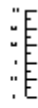
01



02



03



04

5.5.20 Textfelder



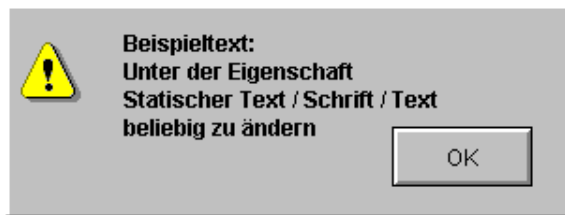
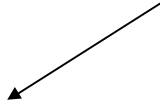
Ausgabertext



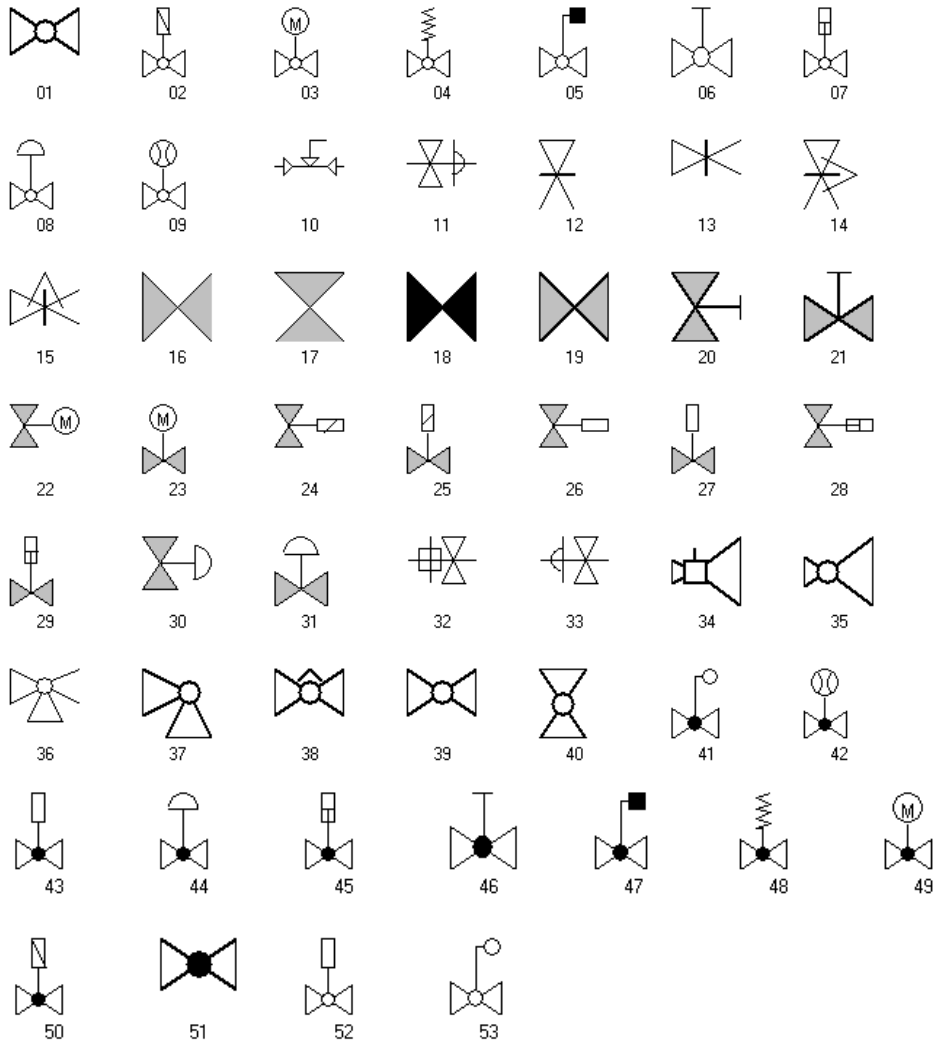
Paßwortfehler



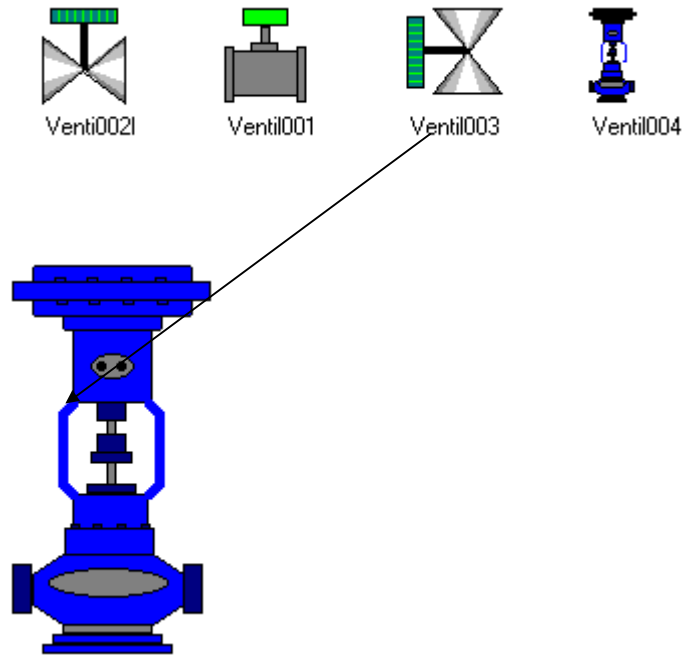
Siemens Wincc



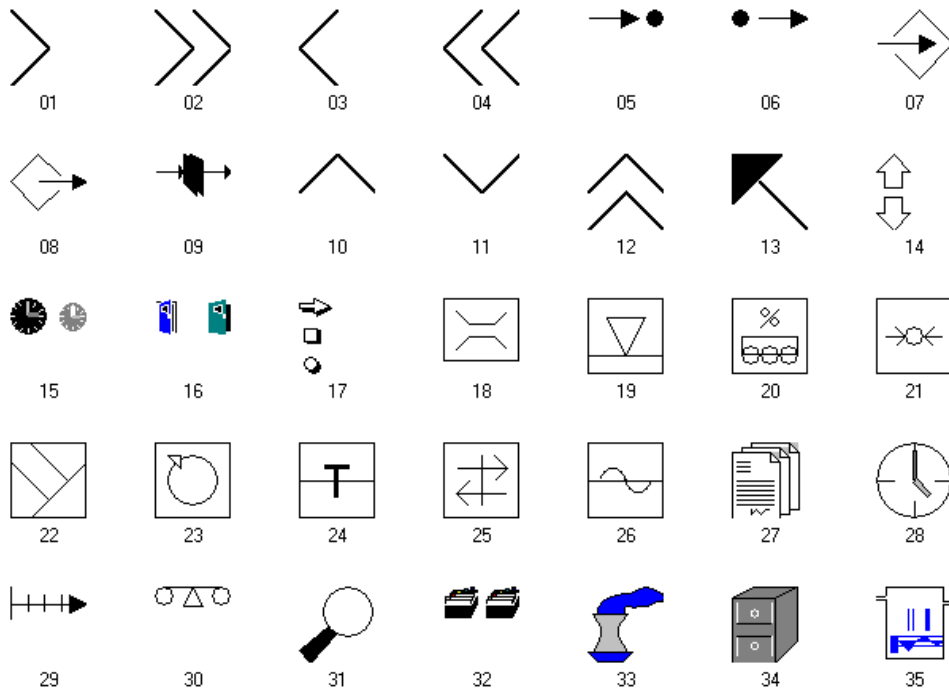
5.5.21 Ventile



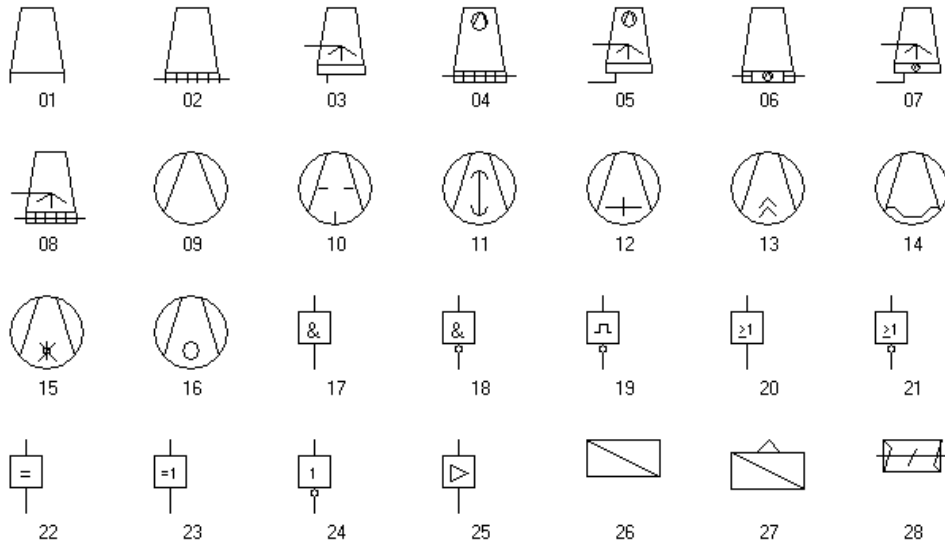
5.5.22 Ventile 3D



5.5.23 Verschiedenes1



5.5.24 Verschiedenes2



Index

A

- Abfragen
 - Ereignis am Objekt, 3-36
 - Tastaturcode, 3-79
- Abkürzungen, 4-43
- Abschalten
 - Runtime-Cursor, 3-80
- Abwicklung, 1-2, 3-1
- Access
 - WinCC Daten lesen, 5-8
- ActiveX, 2-2, 2-3, 3-51, 3-100
- Addition, 4-25
- Adressierung
 - Indirekt, 3-5, 3-93
- Aktionen, 4-7
 - Aktualisierung im Bild, 3-23
 - Aktualisierungszyklen, 3-26
 - Ändern, 4-6
 - Anwahl, 3-30
 - Bei Bildanwahl, 5-3
 - Editor für, 4-8
 - Erstellen, 4-3, 4-11
 - Exportieren, 4-15
 - Festlegen, 3-9
 - Importieren, 4-15
 - In WinCC, 4-4
 - Testen, 4-13
 - Übernehmen, 3-61
 - Verzeichnis, 3-43
 - Wiederverwenden, 4-6
- Aktualisierung
 - Arten der, 3-26
 - Möglichkeiten der Einstellung, 3-23
 - Zyklen, 3-18
- Alarm
 - Allgemeines zur Festlegung, 3-20
 - Dynamisieren, 4-7
 - Im Bedienkonzept, 3-15
- Alphacursor, 3-15, 3-79
- Analogwerte
 - Darstellen, 3-11
 - Uhrzeit, 3-100
- Ändern
 - Anwenderzyklus, 3-25
 - Bildzyklus, 3-31
 - Fensterzyklus, 3-31
 - Mit Variablentrigger, 3-27
 - Projekteigenschaften, 3-52
 - Projektfunktionen, 4-6
 - Standardfunktionen, 4-5
 - Trigger, 3-30
 - Von Zuständen, 3-35
- Anfordern
 - Daten, 3-23
 - Daten vom Datenmanager, 3-26
- Anforderungen, 1-1, 3-10
- Anlagenbild
 - Dynamisieren, 3-36
 - Mauslos bedienen, 3-74
- Anlauf
 - Automatischer, 3-45
 - Systemmeldungen beim, 3-39
- Anlaufliste, 4-12
- Anlegen
 - Datei mit Script, 4-58
 - Meldestruktur, 3-72
 - Strukturvariable, 4-71
- Anpassung
 - Bildaktualisierung, 3-23
 - Daten für Variablenimport, 3-67
 - Rechnereigenschaft, 3-52
- Anschluß
 - An den Prozeß, 3-18
 - USV, 3-48
- ANSI, 4-3
- Anwahl, 1-3, 3-15, 3-28, 3-30, 3-82
- Anweisung
 - Bedingte, 4-42, 4-51
- Anwender, 2-3, 3-10
 - Anwenderdatensätze, 2-3
- Anwenderarchiv, 3-56, 3-102
 - Dynamisieren, 4-7
 - Übernehmen, 3-72
- Anwenderbibliothek, 3-78
- Anwenderobjekt, 3-94, 3-96, 3-99, 3-100
 - In Bildbausteintechnik, 3-90
 - Übernehmen, 3-59
 - Wizard, 3-94
- Anwenderzyklus, 3-25, 3-33
- API, 2-4, 3-61, 3-62, 3-102
- Applikation, 3-23
 - Eigene, 2-2
 - Fremde einbinden, 2-3
 - Schließen, 3-46
 - Schnittstelle (API), 3-62
- Applikationsfenster, 2-3, 3-57, 3-81, 3-85
- Arbeitsbereich, 3-14
- archivieren, 2-2
- Archivierung, 1-1, 3-2, 3-49
 - Archivierungszeiten, 3-23
 - Projektieren, 3-54
- Arithmetik, 4-25
- Array, 4-36

- ASCII, 4-17
- Aufbau, 1-2, 3-98
 - Bedienkonzept, 3-16
 - Bildaufbau, 3-23, 3-90
 - C Struktur, 4-67
 - Firmenstandard, 3-2
 - Funktionen, 4-4
 - Projektierhandbuch, 1-2
 - Textlisten, 3-65
 - Variablenhaushalt, 3-5
 - Verbindungen, 3-99
 - Verbindungsaufbau, 3-103
 - Verzeichnisstruktur, 3-21
 - WinCC, 2-2
- Aufgaben, 1-1
 - Projektteam, 3-54
- Auflösung
 - Bildschirm, 3-13
- Aufruf
 - Funktionen, 4-5
- Ausblenden
 - Von Informationen, 3-11
- Ausführen
 - Variablen Import- Export, 3-64
 - Von Scripten, 3-32
- Ausgabe
 - Im Diagnosefenster, 4-2
 - Normiert, 5-2
 - Testausgabe, 4-14
- Auslagern
 - Meldungen, 3-40
 - Variablen, 3-63
- Auswahl
 - Bilder für Zustandsanzeige, 3-57
 - Der Variablen, 3-5
 - Meldungen, 3-20
 - Trigger, 3-23
- Autorisierung, 3-51
- Autostart
 - Eintrag im Ordner, 3-45

- B**
- Basic, 2-2, 3-101
- Basic Process Control, 3-16, 3-78, 3-85
- Basis
 - Basisprojekt, 3-54, 3-72
 - Für Import, 3-63
- Bausteine
 - Bildbausteintechnik, 3-90
 - Vorprojektiert, 3-59
- Bedienbar
 - Bedienobjekte, 3-79
 - Funktionen mit Zugriffsschutz, 3-47
 - Kurvenfenster, 3-85
 - Meldefenster, 3-81
- Bedienkonzept, 3-15
- Bedienoberfläche, 2-3, 3-2
 - Festlegen, 3-10
- Bedienobjekte, 3-79
- Bedienung, 1-3, 3-12, 3-15
 - Bedienreihenfolge, 1-3
 - Ereignisgesteuert, 3-18
 - Fehlbedienung, 3-10
 - Kurvenfenster, 3-85
 - Mauslos, 3-74
 - Über Funktionstasten, 3-75
 - Über Tastatur, 3-35, 3-78
- Bedingte
 - Anweisungen, 4-42
 - Case Anweisung, 4-53
 - If Anweisung, 4-51
- Bedingung
 - In Schleifen, 4-42
- Beenden
 - Eingabe, 3-81
 - Von WinCC, 3-48
 - WinCC, 3-45
- Begriffe, 2-5
- Beispielprojekt, 1-1, 3-54
 - Bildschirmauflösung, 3-13
 - C Kurs, 4-2
 - Sprache, 3-3
- Benutzer
 - Berechtigung, 3-104
 - Gruppen, 3-19, 3-73
 - Rechte, 3-19
 - Übernahme der Rechte, 3-73
 - Zeitzyklen benutzerdefiniert, 3-24
- Bereich
 - Arbeitsbereich, 3-14
 - Bildschirm, 3-12
 - Tastebereich, 3-17
 - Wertebereich, 3-37, 4-16
 - Zeitbereich, 3-85
- Betriebssystem, 2-2, 3-18, 3-46, 3-48
- Bibliothek
 - Projekt, 3-50, 3-59, 3-60
 - Standard, 3-40
 - Text, 3-68
- Bild
 - Aktualisierung, 3-23
 - Anwahl, 3-15, 3-57, 3-82, 3-86, 3-100
 - Aufbau, 3-23, 3-90
 - Bausteintechnik, 3-90
 - Bildfenster Aufbau, 3-98
 - Bildfensterinhalt, 3-57
 - Größe der, 3-13
 - Hierarchie, 3-78

- Informationen im Bild, 3-10
 - Meldebild, 3-82
 - Namen, 3-7
 - Objekt, 3-35
 - Übernahme von, 3-55
 - Volumen reduzieren, 4-6
 - Wechsel, 3-75
 - Zyklus, 3-25
 - Zyklus ändern, 3-31
 - Bildschirmauflösung, 3-13
 - Bitmap
 - Übernehmen, 3-57
 - Bitmeldeverfahren, 3-20
 - Bitoperationen, 4-33
 - Boolsche Algebra, 4-27
 - Bussystem, 3-18
- C**
- Callback-Funktion
 - zur Rückmeldung des Status eines Schreibauftrags, 5-64
 - zur Übergabe gelesener Prozeßwerte, 5-64
 - C-API, 2-4
 - ChnStart, 5-62
 - Client, 2-4
 - Coros, 3-69
 - Cursor
 - Alpha, 3-15, 3-79
 - Schalt, 3-15, 3-79
- D**
- Dateien
 - Dateioperationen in C, 4-57
 - Im Projektverzeichnis, 3-42
 - Im Standardverzeichnis, 3-39
 - Daten
 - Ablage, 3-4, 3-21
 - Aktualisierung, 3-18
 - Anfordern, 3-23
 - Datenbestand, 2-4
 - Datenhaltung, 2-3
 - Datentyp, 4-35
 - Importieren, 3-49
 - In der Datenbank, 3-40
 - Projektierungsdaten, 2-3
 - Sicherung, 3-50
 - Trennung, 3-42
 - Übernahme, 3-54
 - Übernahme von S5 oder S7, 3-62
 - Datenbank, 2-3, 2-4
 - Abfragesprache, 2-4
 - Änderung, 3-54
 - Rekonstruktion, 3-41
 - Selektion in, 5-12
 - Sicherung, 3-49
 - Zugriff mit Access, 5-8
 - Zugriff mit Excel, 5-5
 - Zugriff mit ISQL, 5-9
 - Zugriff mit Scope, 5-10
 - Datenexport
 - Über C Aktion, 5-11
 - DDE, 2-2
 - Decrementieren, 4-25
 - Default
 - Einstellungen, 3-3
 - Sprache, 3-40
 - Trigger, 3-34
 - Verzeichnis, 3-4
 - Definition
 - C Strukturen, 4-66
 - Schreibweise bei Zeigern, 4-35
 - Deklaration, 4-11, 4-67
 - Diagnose, 4-2, 4-14
 - Datei für Import, 3-63
 - Dateien, 3-39
 - Scope, 5-4
 - Dialog
 - Konfiguration, 3-37, 3-57
 - Konfigurations, 3-23
 - Direktverbindung, 3-23, 3-36, 3-78, 3-92, 3-98
 - Division, 4-25
 - DLL, 4-1, 4-7
 - Dokumentation, 1-1, 1-2, 3-2
 - Dynamic-Wizard, 3-23, 3-29, 3-36, 3-90, 3-94
 - Dynamik-Dialog, 3-24, 3-27, 3-36
 - Dynamische
 - Änderung der Objekteigenschaften, 3-35
 - Instanz, 3-96
 - Verbindung der Variablen, 3-93
 - Dynamisierung
 - Anwenderobjekt, 3-95
 - Art, 3-27
 - Arten, 3-23
 - Eigenschaften, 3-35
 - Ereignisse, 3-36
 - In WinCC, 3-35
 - Objekte, 3-36
 - Projektieren, 3-37
- E**
- Editor, 4-5, 4-8
 - Eigenschaft
 - Am Objekt Dynamisieren, 3-26
 - Funktionskopf an der, 4-4

- Objekt, 3-24
- Eigenschaften einer Kanal-Unit
 - Client-Funktionalität, 5-65, 5-73
 - Diagnosemöglichkeiten, 5-65, 5-73
 - Editieren der Kanal-Properties, 5-65, 5-73
 - Eigene Lebenszeichenüberwachung, 5-65, 5-73
 - Eigene Wiederanlaufanzeige, 5-65, 5-73
 - Eigene Zyklusverwaltung, 5-65, 5-73
 - Keine Registrierung von WinCC-Variablen, 5-65, 5-73
 - Online-Registrierung logischer Verbindungen, 5-65, 5-73
 - Online-Registrierung von WinCC-Variablen, 5-65, 5-73
 - Prozeßwerte in INTEL Bytereihenfolge, 5-65, 5-73
 - Reentrant, 5-65, 5-73
 - Schreiben auf Bit-Adressen, 5-65, 5-73
 - Schreiben auf Byte-Adressen, 5-65, 5-73
 - Uhrzeit-Master, 5-65, 5-73
 - Uhrzeit-Slave, 5-65, 5-73
 - Zugriff auf Remote-Variable, 5-65, 5-73
- Einbinden
 - Von OCX, 3-90
- Einblenden
 - Sourcecode, 4-78
- Eingabe
 - In EA-Feld, 3-79
 - Mittel, 3-15
 - Normiert, 5-2
 - Über Tastatur, 3-74
- Einlesen
 - Meldungen, 3-69
 - S5 7 S7 Variablen, 3-61
- Einplatz
 - System, 3-13, 3-48, 3-54
- Einschränkung
 - Bei Datenübernahme, 3-56
 - Bei Onlineprojektierung, 3-102
 - Bildname, 3-7
 - Variablenname, 3-6
- Empfehlung
 - Für Aktualisierungszyklen, 3-27
- EN 60073, 3-17
- Endlosschleifen, 4-43
- Entwicklung, 2-2
- Entwicklungsumgebung, 3-43, 3-101, 4-3
- Ereignis
 - Dynamisieren, 3-36
 - Funktionskopf am, 4-4
 - Gesteuert, 3-18
 - Trigger, 3-23, 3-32, 3-33
- Ergebnis
 - Ausgeben, 4-17
 - Dynamisierung, 3-37
- Excel, 3-6, 3-21, 3-64, 3-66, 3-69, 3-72
 - Daten mit MSQuery lesen, 5-5
- Export, 3-6, 3-41, 3-49, 3-60, 3-63
 - Aktionen, 4-15
 - Funktionen, 4-15
- Extension
 - Für Aktionen, 3-61
 - WinCC Dateien, 3-42
- F**
- Farb
 - Änderung, 4-76
 - Definition, 3-17
 - Erkennung, 3-11
 - Farben im Projekt, 3-11
- Farbtabelle, 5-14
- Fclose, 4-58
- Fehler
 - Bedienfehler, 3-42
 - Fehlermeldung, 3-52
 - Hinweise, 4-13
 - Suche, 3-39, 4-6
 - Syntaxfehler, 4-13
- Felder, 4-36, 4-67
- Fenster
 - Diagnose, 4-2, 4-14
 - Fehlerfenster, 4-13
 - Fensterwechsel, 3-82
 - Zyklus, 3-25, 3-26, 3-31
- Fgets, 4-64
- FILE, 4-57
- Fließkommazahlen, 4-16
- Float, 4-16
- Fprintf, 4-58
- Fscanf, 4-58
- Funktionalität, 1-1
 - Bildbausteine, 3-96
 - Toolbar Alarm Logging, 3-85
 - Toolbar Tag Logging, 3-87
- Funktionen
 - API, 3-61
 - Aufbau, 4-4
 - Bedien, 3-19
 - Editor, 4-8
 - Erstellen, 4-11
 - Exportieren, 4-15
 - Importieren, 4-15
 - In WinCC, 4-4
 - Interne, 3-26, 4-6
 - Kopf, 4-4
 - Projekt, 3-43, 4-6
 - Projektfunktionen übernehmen, 3-73

- Rumpf, 4-5
- Standard, 3-39, 3-73, 4-5
- Testen, 4-13
- Zugriffsschutz, 3-47

Funktionstaste, 3-74

G

Generieren

- Header neu, 3-57, 3-73, 3-99
- Header neu erstellen, 3-53

Grafik

- Bibliothek, 3-92
- Tools, 3-40

Grundrechnungsarten, 4-29

Gruppe

- Benutzergruppe, 3-19, 3-73
- Benutzerrgruppen, 3-57
- Variablengruppe, 3-5

H

Hardcopy, 3-88

Header

- Generieren, 4-11

Hierarchie

- Bedienung, 3-15
- Bei Anlagen, 3-75
- Bild, 3-78

HMI, 1-1, 2-2, 2-5, 3-18, 3-45

Hotkey, 3-75, 3-76, 3-82, 3-89

I

IF, 4-43, 4-51

Import, 3-6, 3-49, 3-61, 3-63

- Aktionen, 4-15
- Funktionen, 4-15

Incrementieren, 4-25

Information

- Ausblenden, 3-11
- Finden, 3-4
- Im Bild, 3-10

Informix, 2-4

Ingres, 2-4

Inhalt, 1-2

- Funktionsrumpf, 4-5
- Projektverzeichnisse, 3-42
- Zeiger, 4-35

Initialisierung einer Kanal-Unit

- Parameter für den WinCC-Datenmanager, 5-64

Installation

- Tools, 3-40
- USV, 3-48
- WinCC, 3-51
- WinCC Installationspfad, 3-39

Instanz, 3-96

- Anlegen, 3-96

Integer, 4-17

Interne

- Funktionen, 3-26, 3-39

K

Kanal

- DLL, 3-51, 3-62, 3-67
- S7 PMC, 3-103

Klammer, 4-5, 4-42, 4-67

Kommunikation, 3-18, 3-56

- Einfluß auf Aktualisierung, 3-26
- Online Ändern, 3-103
- Prozeß, 3-23
- Schnittstellen, 3-51
- Zwischen einzelnen Tasks, 3-27

Konfiguration

- Dialog, 3-23, 3-28, 3-37, 3-51, 3-57, 3-94

Konvention, 1-3

Konzept, 2-4

- Bedienung, 3-12, 3-15
- Datensicherung, 3-48
- Prototypenbilder, 3-91

Kopplung

- Seriell mit CP525, 5-13

Kurven

- Fenster Bedienung, 3-82, 3-85
- Rahmen übernehmen, 3-56

L

Laufwerk

- Zur Datensicherung, 3-49

Laufzeit, 3-95, 3-100

- Daten, 3-43
- Dynamisierung zur, 3-35

Lizenzprüfungen, 3-39

Log

- Dateien, 3-39, 3-44

Logik, 4-25

- Vergleich, 4-25

Login, 3-89, 3-104

Logoff, 3-104

Logout, 3-89

Lösung

- Automatischer Projektanlauf, 3-45
- Lösungsansatz, 2-2

Lösungsvorschlag, 1-1
 Lösungsweg, 1-1

M

Maus

Aktion für Hotkey, 3-76
 Dynamisierung Ereignisse, 3-36
 Mauslose Bedienung, 3-74

Mehrplatz

System, 3-13, 3-48, 3-51, 3-54

Melde

Meldeklasse, 3-17, 3-70
 Meldeliste, 3-81
 Meldesystem, 3-17

Meldefenster, 3-56, 3-81, 3-86

Meldungen, 2-3

Bitmeldeverfahren, 3-20
 Einlesen, 3-69
 Farben festlegen, 3-17
 Fehler, 4-13
 Meldebild, 3-82
 Meldedatei, 3-70
 Meldeverfahren, 3-20
 Sicherung, 3-49
 Systemmeldungen, 3-39
 Übernehmen, 3-68, 3-69
 Zeitfolgerichtiges Melden, 3-20

Meßwert

Aktualisierung, 3-18
 Archivierung, 3-23
 Auslagern, 3-40
 Erfassung, 3-54
 Übernehmen, 3-72

Modularität, 2-2

Module

Einbinden, 2-2

Multiplikation, 4-25

N

Netzwerk, 3-49, 3-51

O

OCX, 2-3, 3-51, 3-81, 3-100

Bildbaustein, 3-90
 Registrierung, 3-41, 3-101

ODBC, 2-2

OLE, 2-2

Verbindungen, 3-51

Online

Projektierung, 3-102

Operatoren, 4-25

Optionen, 3-16, 3-40, 3-42, 3-78, 3-85

Oracle, 2-4

P

Parameter

Default, 3-43
 Diagnose, 3-103
 Für Bildnamen, 3-7
 Für Bildschirmauflösung, 3-13
 Für printf, 4-18
 Für Projektnamen, 3-4
 Für Variablennamen, 3-6
 Shutdown, 3-48
 Verbindungs, 3-62

Paßwort, 3-19, 3-47, 4-6

Performance, 1-1, 3-2, 3-18, 3-36, 3-54, 4-1

Plattform, 2-2

Pointer, 4-35

Printf, 4-2, 4-14, 4-17

Programm

Autostart, 3-45
 Eigene, 3-61
 Für Datenbank, 3-40
 Tools, 3-6
 Zur Datenübernahme, 3-62
 Zusatzprogramme, 3-51

Programmierschnittstelle, 2-4

Projekt

Automatischer Anlauf, 3-45
 Beispielprojekt, 3-3, 4-1
 Bibliothek, 3-60, 4-15
 Bibliothek übernehmen, 3-59
 Funktion erstellen, 4-11
 Funktionen, 4-6
 Kopieren, 3-51
 Name, 3-4
 Sicherung, 3-49
 Tips zur Realisierung, 3-21
 Übergreifende Funktionen, 3-40, 4-5
 Umgebung, 3-42
 Verzeichnis, 3-42, 4-12
 Wartungsfreundlich, 3-26

Projektierung

Daten der, 3-42
 Dynamisieren, 3-37
 Festlegungen für, 3-2
 Modus, 3-45

Projektierungsdaten, 2-3

Prototyp, 3-90, 3-91, 3-94, 3-96

Prozeß

Anschluß, 3-18
 Bedienung, 3-12

- Hierarchie der Bedienung, 3-15
 - Kommunikation, 3-23
 - Variablen, 3-93
 - Prozeßdaten, 2-4, 3-23
 - Archivieren, 2-2
- Q**
- Quittierung
 - Meldungen, 3-81
- R**
- Radiobox
 - Bedienung über Tastatur, 3-81
 - Rechner
 - Einstellungen, 3-46
 - Hotkey, 3-82
 - Typ ändern, 3-102
 - Verzeichnis für, 3-43
 - Referenz
 - In Bildbausteinen, 3-92, 3-100
 - In Bildern, 3-56
 - Textreferenz, 3-68
 - Registrieren
 - OCX, 3-41, 3-101
 - OLE, OCX, 3-51
 - Rekonstruktion
 - Datenbank, 3-41
 - Richtungstasten, 3-81, 3-83
 - Rückgabewert, 4-4, 4-55
 - Runtime
 - Bedienung im, 3-79
 - Cursor, 3-80
 - Dynamisierung, 3-35
 - Einschränkung bei Onlineprojektierung, 3-102
 - Einstellungen Für, 3-46
 - Festlegungen für, 3-2
- S**
- Schaltcursor, 3-15, 3-79
 - Schleifen, 4-42
 - Do while, 4-42
 - For, 4-43
 - While, 4-42
 - Schnittstelle
 - API, 3-62
 - Schrift
 - Art, 3-2
 - Farbe, 3-17
 - Größe, 3-2, 3-13
 - Scope, 5-4
 - Zugriff auf WinCC Datenbank, 5-10
 - Scripte, 3-9
 - Ausführen, 3-32
 - Editoren für, 4-8
 - Entwicklungsumgebung, 4-3
 - Für Wizard, 3-40
 - Sprache, 4-1
 - Syntax, 4-3
 - Shutdown, 3-48
 - Sicherung, 3-42
 - Konzept für, 3-48
 - WinCC Daten, 3-49
 - SmartTools, 3-6, 3-40, 3-51, 3-63
 - Sonderzeichen, 3-7, 4-17
 - Sourcecode
 - Einblenden, 4-78
 - Speicher, 4-36, 4-67
 - Sprache, 3-68
 - C, C++, 4-35
 - Übernehmen, 3-68
 - SQL, 2-3, 2-4
 - Datenbank, 3-46
 - Programmierung, 3-62
 - Tools, 3-40
 - Standard
 - C, 4-3
 - Einstellungen, 3-23, 3-24
 - Funktionen, 3-50, 4-5
 - Inhalt WinCC Standardverzeichnis, 3-39
 - Installationspfad, 3-39
 - Tasten, 3-77
 - Zyklen, 3-27
 - Start
 - Bild, 4-1
 - WinCC automatisch, 3-45
 - String, 4-36, 4-41
 - Struktur
 - Bildnamen, 3-7
 - Datenbank, 3-54
 - Für Datenablage, 3-21
 - Für Meldungen anlegen, 3-72
 - Variablen, 3-5
 - WinCC Projektverzeichnis, 3-42
 - WinCC Systemverzeichnis, 3-39
 - Strukturen, 4-66
 - Subtraktion, 4-25
 - Switch, 4-44
 - Sybase, 2-3, 3-46
 - Symbol
 - Ablage der, 3-40
 - Exportieren, 3-60
 - Für Dynamisierungsart, 3-35
 - Übernahmen, 3-57
 - Syntax, 4-3

- Fehler, 4-13
- Meldungen, 4-13
- System
 - Automatischer Anlauf, 3-45
 - Belastung, 3-23, 3-27, 3-34
 - Betriebssystem, 3-18
 - Einplatz, 3-13
 - Einzelplatz, 3-54
 - Layout, 3-50
 - Mehrplatz, 3-13, 3-51, 3-54
 - Meldungen, 3-39
 - Module, 2-2
 - Plattform, 2-2
 - Software, 3-51
 - Umgebung, 3-39
 - Variablen, 3-6, 3-56

T

- Tabellenfenster, 3-56, 3-85
- Task
 - Taskleiste, 3-46
 - Taskwechsel, 3-26
- Tastatur
 - Bedienung, 3-15, 3-35, 3-74
 - Funktionstasten, 3-77
 - Hotkey, 3-88
- Tasten
 - Bedientasten, 3-57
 - Funktionstasten, 3-74
 - Hotkey, 3-76
 - Im Kurvenfenster, 3-85
 - Im Meldefenster, 3-83
 - Projektierte, 3-68
 - Tastenbelegung, 3-81
 - Tastenbereich, 3-12, 3-17
 - Tasteneinstellungen, 3-80
 - Tastenkombination, 3-45
- Template
 - Bilder, 3-96
 - Meldefenster, 3-82
- Text
 - Darstellung, 3-11
 - Erkennung am Bildschirm, 3-11
 - Listen, 3-65
 - Mehrsprachige, 3-68
 - Textlibrary, 3-104
 - Textlisten, 3-61, 3-81
- Toolbar
 - Alarm Logging, 3-81, 3-85
 - Tag Logging, 3-85
- Tools
 - Datenbank, 3-49
 - Grafik, 3-40

- OCX, 3-51
- Sprachen, 3-68
- SQL, 3-40
- Variablen Import- Export, 3-6, 3-63
- WinCC, 3-40
- Tooltip, 3-14
- Transaktionsgesichert, 2-3
- Trigger, 4-7
 - Bildaktualisierung, 3-23
 - Im Dynamik-Dialog, 3-30
 - In C-Aktionen, 3-30
 - Variablen, 3-23
 - Variablentrigger, 3-27
 - Zeitgesteuert, 3-33

Ü

- Übersichtsbilder, 3-15
- Umgewandlung
 - Dynamik-Dialog in C-Aktion, 4-1
- Umlaufpuffer
 - Für Bilder, 3-15
- Umwandlung
 - Einplatz - Mehrplatz, 3-54
- UNIX, 2-4
- USV, 3-48

V

- Variablen
 - Anbindung, 3-37
 - Archivvariablen, 3-87
 - Aufbau der Exportierten Listen, 3-65
 - Auslagern, 3-63
 - Festlegen, 3-5
 - Import- Export, 3-49
 - In C, 4-16
 - In Scripten, 3-6
 - Informationen im Bild Darstellen, 3-91
 - Meldung über fehlende, 3-39
 - Name, 4-16
 - Referenzen in Bildern, 3-56
 - Simulation, 3-44
 - Trigger, 3-23, 3-25, 3-27, 3-33
 - Typen, 4-16
 - Übernehmen, 3-61
 - Von S5 / S7 übernehmen, 3-62
 - Wertebereich, 4-17
 - Zulässige Namen, 3-6
- VDE 0199, 3-17
- Verbindung
 - Aufbauen, 3-103
 - Indirekt, 3-90

- Logische, 3-62
 - Mit Prozeßvariablen, 3-99
 - Mit Variablen, 3-93
 - Neue, 3-90
 - Verbindungsliste, 3-66
 - Verzeichnis
 - Daten im WinCC Verzeichnis, 3-40
 - Für Autostart, 3-45
 - Projektbibliothek, 3-60
 - Projektverzeichnis, 3-4
 - Projektverzeichnis von WinCC, 3-42
 - Struktur von WinCC, 3-21, 3-39
 - Tools für WinCC, 3-63
 - Visual
 - Basic, 2-2, 3-101
 - C, 4-13
 - C++, 2-2, 3-101
 - Visualisieren, 2-2
 - Visualisierung
 - Station, 3-23
 - Voraussetzung, 1-2
 - Vorlage
 - Für eigene Projekte, 3-3
 - Vorschriften
 - Bei der Datenübernahme, 3-67
- W**
- Wahrheitstabelle, 4-27
 - While, 4-42
 - WinCC
 - Aktion übernehmen, 3-61
 - Aktionen, 3-9
 - Aktualisierungszyklus, 3-24
 - Alarmierung, 3-20
 - API, 3-61
 - Aufbau, 2-2, 2-3
 - Automatischer Anlauf, 3-45
 - Bedienkonzept, 3-15
 - Bedienoberfläche, 3-10
 - Beenden, 3-48
 - Datensicherung, 3-49
 - Defaulteinstellung, 3-3
 - Dynamisierung, 3-35
 - Logdateien, 3-39
 - Projekt kopieren, 3-51
 - Projektname, 3-4
 - Projektumgebung, 3-42
 - Scripte, 3-9
 - Standardverzeichnis, 3-39
 - Systemumgebung, 3-39
 - Tools, 3-40, 3-51
 - Variablen übernehmen, 3-61
 - Variablenhaushalt, 3-5
 - Version 1.10, 3-71
 - Verzeichnisstruktur, 3-42
 - Windows, 2-2
 - Windows 95, 3-18
 - Windows NT, 3-18
 - Wizard
 - Coros Meldungen einlesen, 3-69
 - Dateien im Standardverzeichnis, 3-40
 - In Anwenderobjekten, 3-94
 - S5 / S7 Variablen einlesen, 3-61
 - Script übernehmen, 3-102
 - Zur Bildaktualisierung, 3-23
 - Zur Objektdynamisierung, 3-36
 - Wrebuild, 3-41
 - Wunload, 3-40
- Z**
- Zahl, 4-5
 - Dezimal, 4-18
 - Fließkommazahl, 4-16
 - Ganzzahlen, 4-16
 - Zahlenformat, 4-13
 - Zeichen
 - Folge, 4-5, 4-36, 4-67
 - Kette, 4-67
 - Zulässige bei Bildnamen, 3-7
 - Zulässige bei Variablenamen, 3-6
 - Zulässige im Projektnamen, 3-4
 - Zeiger
 - In C, 4-35
 - Zeigerinstrument, 3-11, 3-95
 - Zeitfolgerichtiges
 - Melden, 3-20
 - Zeittrigger, 3-32
 - Zeitzyklus, 3-23, 3-26, 3-33
 - Zielgruppe, 4-1
 - Zugriff
 - Auf Datenbank, 3-49, 3-61
 - Rechte, 3-56
 - Zugriffsschutz, 3-15, 3-19, 3-47
 - Übernahme der Rechte, 3-73
 - Zuordnungslisten, 3-62
 - Zustand
 - Änderung, 3-20
 - Beim Verlassen, 3-45
 - Darstellung von Zuständen, 3-11
 - Zustandsanzeige, 3-57
 - Zykluszeit, 3-26

