Inhaltsverzeichnis

Informationen für Leser 13

1 Die Entwicklung der Steuerungstechnik 18
1.1 Die Anfänge der speicherprogrammierbaren Steuerung 19
1.2 Die SPS lernt zu kommunizieren 22
1.3 Entwicklung von Feldbussystemen 24
1.4 Integration von Anzeige-Systemen in die SPS 25
1.5 Integration von Motion Control in die SPS 27
1.6 Antriebe werden zu vollwertigen Bus-Teilnehmern 30
1.7 PLC und PAC – der Unterschied 31
1.8 Fazit zur bisherigen Entwicklung 32

2 Grundsätzliches zur objektorientierten Programmierung 33
2.1 Basis der objektorientierten Programmierung 33
2.1.1 Geschichte 33
2.1.2 Was ist anders? 34
2.1.3 Was bedeutet Objektorientierung? 35
2.1.4 Objekte und deren Interaktionen 36
2.2 Allgemeine Prinzipien von OOP 38
2.2.1 Objekte 38
2.2.2 Klassen 39
2.2.3 Vererbung 40
2.2.4 Überschreibung 41
2.2.5 Interfaces zur Objekt-Interaktion 42
2.2.6 Zusammenfassung 44
2.2.7 Vorteile der Nutzung von OOP 45
2.2.8 Nachteile der OOP 46
2.3 Tipps zur Klassenbildung 46

3 Objektorientierte Programmierung 49
3.1 Umsetzung von OOP mit SIMOTION 49
3.2 Funktionsbausteine mit Methoden 51
3.2.1 Modularisierung ohne OOP-Erweiterungen 51
3.2.2 Programm und Daten sind getrennt 53
3.2.3 Weiterentwicklung im Softwarelebenszyklus 56
3.2.4 Nachteile der Programmierung ohne OOP-Erweiterungen 57
3.2.5 Erweiterungen zu FBs und deren Zugriffsspezifikation 58
3.2.6 Nutzung von Methoden zur übersichtlichen Programmierung 59
3.2.6.1 Beispiel FB mit Methoden 60

3.2.6.2 Aufrufbeispiel des Funktionsbausteins ... 62
3.2.7 Funktionsbaustein mit Methoden zur Kommandierung ... 63
3.2.7.1 Beispiel des FB mit Kommando-Methoden ... 64
3.2.7.2 Aufrufbeispiel zum FB mit Kommando-Methoden ... 65

3.3 Klassen (CLASS) ... 67
3.3.1 Unterstützte Schlüsselworte für eine Klasse ... 68
3.3.2 Methoden (METHOD) ... 70
3.3.3 Methoden und deren Zugriffspezifikation ... 71
3.3.4 Deklaration von Klasseninstanzen ... 72
3.3.5 Regeln für die Bezeichner in einer Klasse ... 73
3.3.6 Verwendung von Klassen-Methoden ... 73
3.3.6.1 Beispiel CLASS Counter ... 74
3.3.6.2 Nutzung der Methode von CLASS COUNTER ... 75
3.3.6.3 Erweiterung von CLASS COUNTER und Nutzung von THIS ... 76
3.3.6.4 Nutzung der Methoden UP und DOWN ... 77
3.3.7 Klassen und Vererbung ... 78
3.3.7.1 Beispiel Ableitung einer Klasse ... 79
3.3.7.2 Beispiel zur Nutzung der Basis- und abgeleiteten Klasse ... 81
3.3.7.3 Weitere Aspekte zum Methodenaufruf ... 82
3.3.7.4 Beispiel von Basis- und abgeleiteter Klasse in einer Funktion ... 83
3.3.8 Abstrakte Klassen ... 84

3.4 Beispiele zur Nutzung von Ventilen mit OOP ... 85
3.4.1 Beispiel mit 4/3-Wegeventil ... 86
3.4.1.1 Beispiel für eine Klasse für 4/3-Wegeventile ... 87
3.4.1.2 Beispiel für den Aufruf der Ventile ... 88
3.4.1.3 Beispiel mit 4/3-Wegeventil mit Eilgang/Schleichgang ... 90
3.4.1.4 Beispiel einer abgeleiteten Klasse ValveControl43FS ... 91
3.4.1.5 Beispiel für Aufruf der Basisklasse und der erweiterten Klasse ... 92
3.4.1.6 Beispiel Aufruf der erweiterten Klasse mit Basisfunktion ... 93

3.5 Interfaces ... 94
3.5.1 Unterstützte Features ... 95
3.5.2 Prinzipien von Interfaces ... 96
3.5.2.1 Beispiel zur Deklaration von Interfaces ... 97
3.5.3 Repräsentanz der Interfaces im PNV von SCOUT ... 99
3.5.4 Nutzen der Interfaces ... 100
3.5.5 Interfaces als Verweis auf Klassen ... 101
3.5.6 Ventil-Klassen mit Interfaces ... 105
3.5.7 Vereinbarung des Interfaces für Ventil ... 106
3.5.7.1 Beispiel ValveControl43 mit Endschalterüberwachung ... 107
3.5.7.2 Beispiel ValveControl43 mit Error Reporting ... 109
3.5.7.3 Beispiel ValveControl43 mit Test-Error-Reporting ... 114
3.5.7.4 Beispiel Klasse HMIReporting ... 115
3.5.7.5 Beispiel ValveControl43 mit Error Reporting ... 116
3.5.8 Interface für die Neutralisierung von Peripherie-Komponenten ... 118
3.5.8.1 Anbindung von Kameras an die Steuerung ... 118
3.5.8.2 Interface-Definition für eine Kamera-Anbindung ... 124

3.5.9 Interface für neutrale Peripherie-Anbindung (reduziert) ... 125
3.5.9.1 Interface-Definition für neutrale Peripherie-Anbindung ... 127
3.5.9.2 Implementierung in Klassen ... 127
3.5.9.3 Programm der Interface-Definition und Mapping-Tabelle ... 128
3.5.9.4 Programm der Klassenimplementierung und Verwendung ... 129
3.5.9.5 Interface für Eilgang-Schleichgang-Umschaltung ... 131
3.5.9.6 Klassenimplementierung für Eilgang-Schleichgang ... 132

3.6 Weitere Optimierungen in der Klasse Ventil ... 133
3.6.1 Bisherige Realisierung von ValveControl ... 133
3.6.2 Aufbau eines Zustandsautomaten ... 134
3.6.2.1 Beispiel ValveControl43ST – Zustandsautomat über Case ... 136
3.6.2.2 Beispiel ValveControl43ST – Zustandsautomat mit Klassen ... 142

3.7 Abstrakte Klasse für unterschiedliche Antriebe ... 145
3.7.1 Funktionelle Unterschiede der jeweiligen Antriebslösungen ... 146
3.7.2 Klassenmodell zur Anbindung unterschiedlicher Antriebe ... 148
3.7.2.1 Beispiel Abstrakte Klasse „CDrive“ ... 149
3.7.2.2 Beispiel Klasse der direkt einschaltbaren Antriebe ... 150
3.7.2.3 Beispiel Klasse der Stern-Dreieck-geschalteten Antriebe ... 152
3.7.2.4 Beispiel Klasse der drehzahlgeregelten Antriebe ... 153
3.7.2.5 Beispiel Ansteuerung der unterschiedlichen Antriebe ... 158

3.8 Abstrakte Klasse versus Interface ... 160
3.9 OOP öffnet die Welt der Entwurfsmuster ... 161

4 OOP unterstützt modulare Software-Konzepte ... 163
4.1 Zusammenbau von Projekten für reale Maschinen ... 164
4.1.1 Modulgestaltung ... 165
4.1.2 Die Rolle der Software-Entwickler ... 165
4.1.3 Software modularisieren ... 167
4.1.3.1 Equipment-Module erstellen ... 168
4.1.3.2 Software-Design des Equipment-Moduls ... 170
4.1.3.3 Beispiel für die Klasse „CEMPusher“ ... 171
4.1.3.4 Beispiel für den Aufruf des Equipment-Moduls ... 176
4.1.4 Vorbereitung zur mehrfachen Wiederverwendung ... 178
4.1.4.1 Beispiel des neutralisierten Equipment-Moduls ... 178

4.2 Projektgenerator SIMOTION easyProject ... 180
4.2.1 Erweiterung des Projektgenerators um eigene Module ... 184
4.2.2 Bedienung für den Projektgenerator erstellen ... 185
4.2.3 XML-Beschreibung des Equipment-Moduls ... 187

5 Hinweise zum Design und der Entwicklung von Software ... 191
5.1 Ermittlung der Anforderungen ... 191
5.1.1 Aufsetzpunkt Bedienoberflächen ... 192
5.1.2 Aufsetzpunkt Prozessabläufe ... 192
5.1.3 Aufsetzpunkt Elemente des Maschinenbaus ... 193
5.1.4 Bestehende Lösungen ... 194

5.2 Objektorientiertes Design ... 195
5.2.1 Kapselung ... 195
5.2.2 Verantwortung einer Klasse ... 196
5.2.3 Gemeinsamkeiten und Unterschiede von Objekten ... 197
5.2.4 Prinzip der Ersetzbarkeit bei Ableitungen ... 198
5.2.5 Ermittlung der Beziehungen ... 198
5.2.6 SOLID-Prinzipien ... 200

5.3 Wiederverwendbare und leicht wartbare Software ... 200
5.3.1 Wie ist Wiederverwendbarkeit erreichbar? ... 200
5.3.2 Bibliotheken helfen ... 201
5.3.3 Wie sind Module zu entwickeln? ... 202

5.4 Organisatorische und rechtliche Aspekte ... 205
5.4.1 Umstellung muss geplant werden ... 205
5.4.2 Software muss geplant werden ... 206
5.4.2.1 Analyse der bestehenden Programme ... 206
5.4.2.2 Wiederverwendung von Software ... 207
5.4.3 Wiederverwendung und Eigentum der Software ... 209
5.4.3.1 Weitergabe von Software ... 210
5.4.3.2 Übernahme von Software ... 211
5.4.4 „Gute Software“ und objektorientiertes Design ... 212

5.5 Softwaretests sind ein Muss! ... 215
5.5.1 Modultest ... 217
5.5.2 Integrationstest ... 218
5.5.3 Systemtest ... 219
5.5.4 Abnahmetest ... 220

6 Ergänzende Themen zur Softwarestrukturierung ... 222

6.1 I/O-Referenzen ... 222
6.1.1 Deklaration ... 223
6.1.2 Bindung der Referenzen an I/O-Variablen ... 223

6.2 Namensräume ... 225
6.3 Allgemeine Referenzen ... 228
6.3.1 Deklaration und Initialisierung ... 228
6.3.2 Arbeiten mit Referenzen ... 229

7 Referenz zu Erweiterungen von SIMOTION ... 233
7.1 Allgemeine Erweiterungen des Programmiermodells ... 233

7.2 Klassen in SIMOTION ... 234
7.2.1 Konstanten und anwenderdefinierte Datentypen in Klassen ... 234
7.2.2 Benennung von Variablen in Klassen und Methoden ... 235
7.2.3 Methodenaufrufe ... 236
7.2.4 FINAL bei Methoden und Klassen ... 237
7.2.5 Deklaration von abstrakten Klassen und Methoden ... 237
7.2.6 Interface-Implementierung und Klassen-Ableitungen ... 238
7.2.7 Typumwandlungen bei Klassen und Interfaces ... 239

7.3 Instanziierung von Klassen und Funktionsbausteinen ... 241
7.3.1 Anwenderdefinierte Initialisierung von Instanzen ... 241
7.3.2 Vorbelegung von Interface-Variablen ... 243
7.3.3 Anlegen von Klassen- und Funktionsbausteininstanzen ... 243
7.3.4 RETAIN-Daten in Klassen und Funktionsbausteinen ... 244
7.3.5 Arrays variabler Länge ... 245

7.4 Tipps für kompatible und effiziente Software ... 245
7.4.1 Methoden- und Funktionsaufrufe ... 245
7.4.2 Verwendung von Enum-Werten und Konstanten ... 246
7.4.3 Nutzung vordefinierter Namensräume ... 247
7.4.4 Deklaration von Datentypen, Variablen und Methoden ... 248
7.4.5 Aufbereitung von strukturierten Daten zur Datenübertragung ... 249

8 Einführung in SIMOTION ... 252
8.1 Klassische Entwicklung von Steuerungssystemen ... 252
8.2 Neue Steuerungskonzepte erforderlich ... 253
8.3 Technologische Objekte in SIMOTION ... 254
8.4 Drei Hardware-Plattformen ... 256
8.5 Anbindung von Antrieben und Peripherie an SIMOTION ... 257
8.6 Handling-Kinematiken in SIMOTION ... 258
8.7 Das Programmiermodell von SIMOTION ... 258
8.7.1 Die Units von SIMOTION ... 259
8.7.2 Variablenmodell in SIMOTION ... 261
8.7.3 Bibliotheken in SIMOTION ... 264

8.8 Das Engineeringsystem SIMOTION SCOUT ... 265
8.9 Komponenten von SCOUT ... 266
8.9.1 Der Projektnavigator von SCOUT ... 267
8.9.2 Neues Projekt anlegen ... 268
8.9.3 Neues Gerät anlegen ... 270
8.9.4 Hardware-Konfiguration ... 273
8.9.5 Adressliste von SIMOTION ... 275
8.9.6 Achsen anlegen ... 276
8.9.7 Antriebe anlegen ... 281
8.9.8 Bahnobjekte anlegen ... 283
8.9.9 Spracheditoren in SCOUT ... 285
8.9.10 Unterstützung der Programmiersprachen ... 287
8.9.11 Programmqellen (Units) einfügen ... 288
8.9.12 Programme eingeben ... 290
8.9.13 Programme dem Ablaufsystem zuordnen ... 292
8.9.14 Integrierte Testfunktionen ... 293
8.9.15 Testen mit Status Programm ... 294

Hinweis zur Nutzung der Beispielprogramme ... 302
Stichwortverzeichnis ... 303