REVERSE-ENGINEERING DES URSPRÜNGLICHEN SUBSYSTEMS MANUELLE JUSTAGE

Dokument zur Studienarbeit
- Designphase -

Autoren	Thomas Kullmann, Günther Reinecker
Dokumentversion	1.6
Zustand	abgeschlossen
letzte Bearbeitung	06.06.02

Inhalt

I Überblick

Das Dokument ist eine kurze und präzise Beschreibung der ursprüngliche Implementation des Subsystems ‚Manuelle Justage'. Dazu wurde dieses stark formalisiert, für die Layoutkonventionen siehe [6] Die einzelnen Elemente sind fett hervorgehoben und werden jeweils kurz erläutert, wobei auch auf Zusammenhänge untereinander hingewiesen wird. Die Anordnung der Codeelemente in diesem Dokument, ist so vorgenommen, dass sich logisch und intuitiv Sinneinheiten ergeben. Die Reihenfolge der Dokumentation entspricht daher nicht immer der im Quelltext.

Den Kern bildet die Klasse `TAngleControl`, welche sowohl die Kontrolle der Oberfläche, als auch die funktionellen Aspekte zur Steuerung der Antriebe beinhaltet. Da diese von der Basisklasse für modale Dialogfenster `TModalDlg` abgeleitet wurde, ist der Aufbau von `TAngleControl` stark an dieser orientiert. Zum besseren Verständnis der Strukturierung siehe deshalb [2].

Für allgemeine Informationen über die ursprüngliche Manuelle Justage ist die, in einem Reverse Engineering Prozess erstellte, Verhaltensspezifikation (siehe [1]) hinzuzuziehen. Des Weiteren gibt es eine Bewertung des zugehörigen Dialogfensters aus Softwareergonomischer Sicht (siehe [5]).

Der Zugriff, auf die zur Justage benötigten Antriebe erfolgt durch Funktionen eines C-Interface. Es wurde bereits einem Reverse-Engineering Prozess unterzogen. Sehr detaillierte Informationen findet man unter [3]. Außerdem ist die h-Datei (M_LAYER.H) sehr umfangreich und detailliert kommentiert.

Für die Manuelle Justage existieren zwei Ressourcen für das Dialogfenster. Es gibt eine mit deutscher Beschriftung und eine mit Englischer. Da die Ressourcen-IDs für beide die gleichen sind, werden in der Ressourcenübersicht (**II.3 Ressourcen**) nur die deutschsprachigen Elemente aufgelistet.

► Dokumentation des Istzustands

Die Dokumentation bezieht sich auf den Quellcode der unten aufgelisteten Dateien (Stand: 24.04.2002). Nachfolgende Änderungen oder Neuimplementationen, können nicht berücksichtigt werden.

h-Dateien (Deklaration)	cpp-Dateien (Implementation)
• SWINTRAC.H	• M_DLG.CPP

Tabelle 1 „Auflistung der zum Subsystem zugehörigen Dateien" (Quelle: selbst)

II Klasse TAngleControl

Deklaration : MJ_OLD.H
Implementation : MJ_OLD.CPP

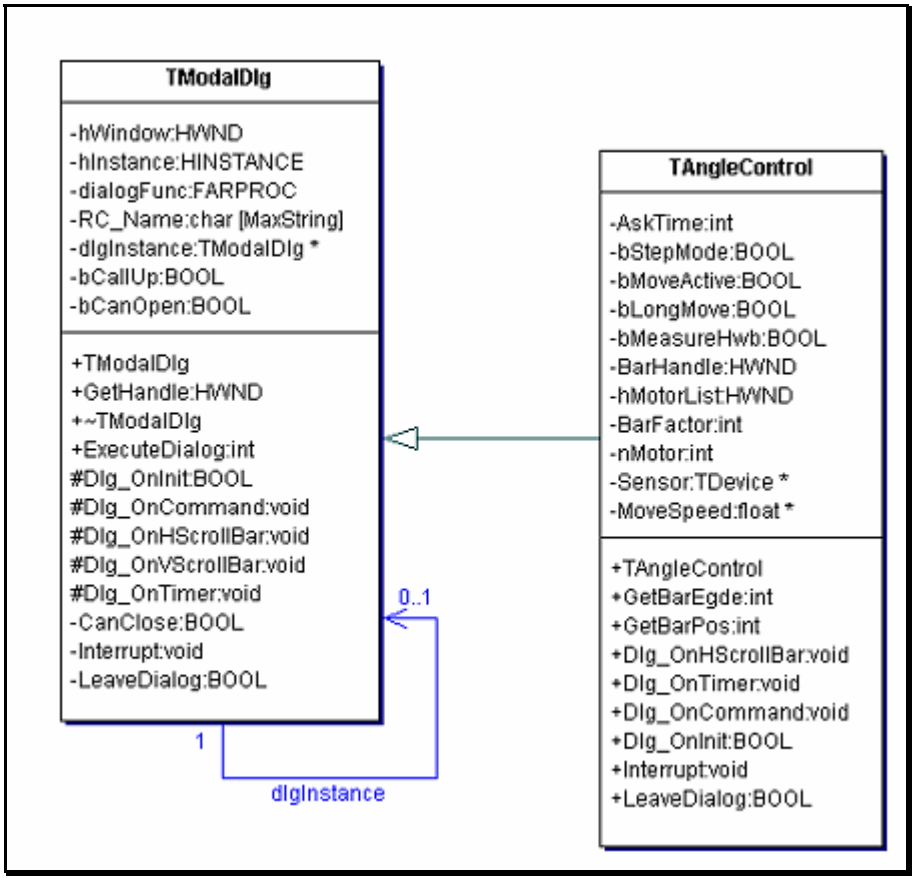

Abbildung 1 „UML-Klassendiagramm der Klasse TAngleControl und ihrer Basisklasse“ (Quelle: Together®, Version 6.0)

II.1 Attribute

▶ **`int AskTime`** **PRIVATE**

ist das Timerintervall, dass bei Methode `SetTimer` zum Starten des Timers verwendet wird; im Konstruktor mit 100 ms initialisiert

▶ **`BOOL bStepMode`** **PRIVATE**

Dieses Flag kennzeichnet, dass das Optionsfeld ‚Schrittbetrieb’ ausgewählt ist. Bei `FALSE` ist das Optionsfeld ‚Fahren’ ausgewählt. im Konstruktor mit `TRUE` initialisiert

▶ **BOOL bLongMove** **PRIVATE**

Dieses Flag kennzeichnet, dass die Positionsangaben (im Eingabefeld ‚Winkel' und in der Bildlaufleiste) beim Aufruf von Methode `Dlg_OnTimer` aktualisiert werden, das funktioniert jedoch nur, wenn auch `bMoveActive == TRUE`. wird verwendet wenn:

- das Dialogfenster betreten wird
- ein neuer Antrieb ausgewählt wird
- wenn die Relative Null gesetzt oder aufgehoben wird
- um den Fortschritt bei Antriebsbewegung im *Fahr* und *Direktbetrieb* anzuzeigen

im Konstruktor mit `FALSE` initialisiert

▶ **BOOL bMoveActive** **PRIVATE**

zu setzendes Flag, wenn sich ein Antrieb in Bewegung befindet, um bei Methode `Dlg_OnTimer` die Bildschirmaktualisierung zu aktivieren (nur in Verbindung mit `bLongMove`); wird im Konstruktor mit `FALSE` initialisiert

es wird nicht unterschieden welcher Antrieb in Bewegung ist → Widerspruch zur Anforderung an das System (siehe [4]: Fehler 6)

▶ **BOOL bMeasureHwb** **PRIVATE**

zu setzendes Flag, wenn die *Halbwertsbreite* gemessen wird; Für die Dauer der Messung sind ein Großteil der Steuerelemente des Dialogfensters gesperrt. wird im Konstruktor mit `FALSE` initialisiert

▶ **HWND BarHandle** **PRIVATE**

ist ein Handle auf die Bildlaufleiste; wird in `Dlg_OnInit` mit dem Handle des Steuerelements `ID_BAR` initialisiert

▶ **HWND hMotorList** **PRIVATE**

ist ein Handle auf die Auswahlliste *Antrieb auswählen*; wird in Methode `Dlg_OnInit` mit dem Handle des Steuerelements `ID_CHOOSEMOTOR` initialisiert

▶ **int BarFactor** **PRIVATE**

ist Skalierungsfaktor zwischen Antriebsposition und Position des Bildlauffeldes, es gilt:

```
BarFactor = (int)(steps / 20000l)+1
```

`Steps` entspricht der Different zwischen maximaler und minimaler Antriebsposition in **Encodersteps**;

wird nur in Methode `GetBarEgde` geschrieben, wenn die Minimalposition neu berechnet wird

Das führt zu Fehlern, wenn zuerst die Position des Bildlauffeldes oder die Maximalposition berechnet wird (weil `BarFactor` noch nicht aktualisiert wurde).

▶ **int nMotor** **PRIVATE**

der Index des derzeit (in der Auswahlliste *Antrieb auswählen*) angezeigte Antriebs; wird in Methode `Dlg_OnInit` mittels `mlGetAxis` initialisiert

▶ **TDevice *Sensor** **PRIVATE**

ist ein Zeiger auf den aktuellen Detektor, der in Methode `Dlg_OnInit` mittels `lpDList->DP()` initialisiert wird; zur Messung der *Halbwertsbreite* verwendet (siehe Methode `TSteering::StartUp`)

▶ **`float *MoveSpeed`** **PRIVATE**

ist die Liste der zuletzt verwendeten Antriebsgeschwindigkeiten; die Länge ist die Anzahl der verfügbaren Antriebe und wird mit Methode `mlGetAxisNumber` ermittelt; wird im Konstruktor dynamisch erzeugt

wird nicht wieder freigegeben

II.2 Methoden

▶ **`TAngleControl( void )`** **PUBLIC**

ist der Standardkonstruktor; ruft den Konstruktor der Basisklasse `TModalDlg`, wenn Symbol `GermanVersion` definiert ist, wird die deutsche Ressource `AngleControl` (sonst `AngleControlEng`) des Dialogfensters angezeigt; Initialisierungen siehe **II.1 Attribute**

▶ **`virtual BOOL Dlg_OnInit( HWND, HWND, LPARAM )`** **PUBLIC**

stellt den Grundzustand des Dialogfensters her: Antriebsliste, Parameter des ausgewählten Antriebs, und Steuerelemente werden (je nachdem welche Antriebe und Funktionen zur Verfügung stehen) freigegeben oder gesperrt; initialisiert die restlichen Attribute, die im Konstruktor nicht initialisiert wurden

▶ **`virtual BOOL LeaveDialog( void )`** **PUBLIC**

wird durch `IDABORT` in der Methode `TModalDlg::Dlg_OnCommand` gerufen; Halbwertsbreitenmessung und Antriebe werden gestoppt

▶ **`int GetBarEgde( int )`** **PUBLIC**

ermittelt die Minimal- (Parameter == LEFT) oder die Maximalposition (sonst) der Bildlaufleiste, indem die *Antriebsposition* skaliert werden (siehe `BarFactor`)

Rechtschreibfehler im Methodennamen, gemeint ist hier bestimmt `GetBarEdge` anstatt `GetBarEgde` → mögliche Fehlerquelle bei Benutzung!

▶ **`int GetBarPos( void )`** **PUBLIC**

berechnet die Position des Bildlauffeldes aus der aktuelle *Antriebsposition* und `BarFactor`

▶ **`virtual void DlgOnHScrollBar( HWND, HWND, UINT, int )`** **PUBLIC**

wird durch Bildlaufleisten-Benachrichtigungsbotschaften ausgelöst; verarbeitet werden `SB_LINEUP`, `SB_LINEDOWN` und `SB_ENDSCROLL` um den Antrieb im *Fahr-* und *Schrittbetrieb* zu steuern; Eingaben werden nur verarbeitet, wenn der Antrieb steht (`bMoveActive == FALSE`) und keine Halbwertsbreitenmessung (`bMeasureHwb == FALSE`) durchgeführt wird.

Fehler: siehe `bMoveActive` und [4]: Fehler 8

▶ **`virtual void Dlg_OnTimer( HWND, UINT )`** **PUBLIC**

Ist die Methode, die bei einem Timerereignis gerufen wird, um das Dialogfenster zu aktualisieren. folgende Zustände werden unterschieden:

- Bewegung beendet und Zielposition erreicht → neue Bewegungsparameter ausgeben
- Bewegung beendet, Zielposition nicht erreicht → Timer neu starten
- Bewegung nicht beendet → Timer neu starten

► virtual void Interrupt() PUBLIC

Wird aufgerufen; wenn das spezielle Tastaturereignis `VK_ESCAPE` ausgelöst wurde oder ein Ende der Bildlaufleiste erreicht ist. stoppt die Bewegung des angezeigten Antriebs (mittels `mStopDrive`) und wartet kurz, bis der Antrieb wirklich gestoppt hat

► virtual void Dlg_OnCommand(HWND, int, HWND, UINT) PUBLIC

Diese Methode verarbeitet fast alle Steuerelementaktionen des Dialogfensters und alle sonstige (interne) Botschaften.

1.TEIL: durch explizit gerufene Botschaften ausgelöst (Überschrift ist Ressourcen-ID der Nachricht)

`cm_ParamSet`

- freigeben und sperren von Steuerelementen im Dialogfenster
- benutzt bei der Initialisierung, bei Halbwertsbreitenmessen, Auswahl eines Antriebs, …
- wird immer in Verbindung mit `cm_MotorInit` aufgerufen

`cm_MotorInit`

- anzeigen der verwendeten Betriebsart, freigeben und sperren einiger Steuerelemente, Ändern der Beschriftung der Steuerelemente, …
- benutzt bei Initialisierung, bei Halbwertsbreitenmessung, Auswahl eines Antriebs, …
- wird immer in Verbindung mit `cm_ParamSet` aufgerufen

`cm_CounterSet`

- `UpdateDisplay` von Objekt `Sensor` aufrufen, zur Aktualisierung des Zählerfensters

`cm_SteeringReady`

- von `TSteering` ausgelöst, um das Ende der Halbwertsbreitenmessung anzuzeigen
- Ausgabe der gemessenen *Halbwertsbreite,* freigeben und sperren der Steuerelemente im Dialogfenster (`cm_ParamSet`)

`cm_MoveButton`

- Flag für *Antrieb in Bewegung* setzen
- Cursor mit der Kennung `IDC_Wait` (Sanduhr) zeigen, Timer starten

`IDOK`

der Nutzer hat die Eingabe in einem Eingabefeld mit [ENTER] abgeschlossen:

- neue *Sollposition* → bewegen des Antriebs zur eingegebenen Position
- neue *Geschwindigkeit* → aktualisieren des Eintrags in `MoveSpeed[]`
- neue *Schrittweite* → Bewegungsparameter *Schrittweite* beim Antrieb setzen und ausgeben
- **sonst → Endlosschleife**

2.TEIL: durch Steuerelemente ausgelöste Ereignisse (Überschrift ist Ressourcen-ID des Elements)

cm_MeasureHWB
- Halbwertsbreitenmessung starten (Messung `Steering::StartMacroExecution`) bzw. stoppen (Methode `Steering::ToggleInterrupt`)
- freigeben bzw. sperren der Steuerelemente im Dialogfenster (`cm_ParamSet`)

cm_SetAngleZero
- setzen der *Relativen Null*, aber nur wenn Antrieb still steht
- Positionsangaben aktualisieren

cm_CancelRelativeZero
- aufheben der *Relativen Null*, aber nur wenn Antrieb still steht
- Positionsangaben aktualisieren

cm_SetFine
- Schnellauswahl des Antriebs ‚Beugung fein', siehe `id_ChooseMotor`

cm_SetTilt
- siehe `cm_SetFine`, für den Antrieb ‚Tilt'

cm_SetCollimator
- siehe `cm_SetFine`, für den Antrieb ‚Kollimator'

cm_RotateMotor
- zum Rotieren der Antriebe innerhalb der Auswahlliste
- **wird im Dialogfenster nicht eingeblendet, ständig nicht sichtbar**

id_ChooseMotor
- ausgelöst durch Auswahl eines Antriebs in der Auswahlliste *Antrieb auswählen*
- `nMotor` speichert den Index des aktivierten Antrieb
- Aufruf von `cm_ParamSet` und `cm_MotorInit`

id_StepMode
- Flag für *Schrittbetrieb* `bStepMode` auf `TRUE` setzen

id_LongMoveMode
- Flag für *Schrittbetrieb* `bStepMode` auf `FALSE` setzen

id_NewAngle
- speichert das Handle auf das Eingabefeld *Sollposition* für `IDOK`

id_AngleWidth
- speichert das Handle auf das Eingabefeld *Schrittweite* für `IDOK`

id_SpeedValue
- speichert das Handle auf das Eingabefeld *Geschwindigkeit* für `IDOK`

II.3 Ressourcen

Ressourcen-ID	Typ	Bezeichnung
cm_MeasureHWB	Schaltfläche	‚Halbwertsbreite messen'
cm_SetAngleZero	…	‚Relative Null setzen'
cm_CancelRelativeZero	…	‚Relative Null aufheben'
cm_SetFine	…	‚Beugung fein'
cm_SetTilt	…	‚Tilt'
cm_SetCollimator	…	‚Kollimator'
cm_RotateMotor	…	‚R' (versteckt)
IDABORT	…	‚Verlassen'
id_ChooseMotor	Auswahlliste	Antriebsliste
id_StepMode	Optionsfeld	‚Schritt-Betrieb'
id_LongMoveMode	…	‚Fahren'
id_Angle	Eingabefeld	‚Winkel'
id_NewAngle	…	‚Neuer Winkel'
id_AngleWidth	…	‚mit D = '
id_SpeedValue	…	‚mit V = '
id_Bar	Bildlaufleiste	

Tabelle 2 „Ressourcen-IDs der Dialogfenster AngleControl und AngleControlEng" (Quelle: selbst)

II.4 Bewertung

Metrik		Kennung (min,max)	Wert
Klasse			
	‚**L**ines **O**f **C**ode' inkl. Leer- und Kommentarzeilen	LOC (0, 1000)	508
	‚**LOC** of **I**mplementation'*	LOCI	493
	‚**LOC** of **D**eclaration'*	LOCD	25
	‚**N**umber **O**f **A**ttributes'	NOA (0, 30)	11
	‚**N**umber **O**f **O**perations'	NOO (0, 50)	8
	‚**N**umber **O**f **M**embers' Attribute + Methoden	NOM = NOA + NOO	19
	‚**N**umber **O**f **C**onstructors'	NOCON (0, 5)	1
	‚**N**umber **O**f **O**verridden Methods'	NOOM (0, 10)	4
	‚**P**ercentage of **Priv**ate **M**embers'	PPrivM	55
	‚**P**ercentage of **Prot**ected **M**embers'	PProtM (0, 10)	0
	‚**P**ercentage of **Pub**lic **M**embers'	PPubM	45
	‚**W**eighted **M**ethods **P**er **C**lass'	WMPC1 (0, 30)	**51**
Attribute			
	‚**A**ttribute **C**omplexity'	AC	75
Methoden			
	‚**M**aximum **N**umber **O**f **P**arameters'	MNOP (0, 4)	**4**
	‚**C**yclomatic **C**omplexity'	CC	51
Kommentare			
	‚**N**umber **O**f **C**omments'*	NOC	97
	‚**T**rue **C**omment **R**atio'	TCR (5, 400)	18

Tabelle 3 „ausgewählte Metriken der Klasse `TAngleControl`" (Quelle: Together® ‚Version 6.0)

* Diese Metriken sind nicht Bestandteil von Together, sondern wurden manuell ermittelt.

Die inhaltlichen Fehler sind in [4] zusammengestellt.

III Attribute

► **`extern BOOL bManualMovesCorrected`**

Definition, um die in M_MAIN.CPP deklarierte Antriebsliste auch in MJ_OLD.CPP benutzen zu können

► **`extern TSteering Steering`**

Definition, um das in M_STEERG.CPP deklarierte Objekt (zur Ausführung des Makros „Halbwertsbreite messen") auch in MJ_OLD.CPP benutzen zu können

► **`extern LPDList lpDList`**

Definition, um die in COUNTERS.CPP deklarierte Detektorliste auch in MJ_OLD.CPP benutzen zu können

IV Anhang

IV.1 Verwandte Dokumente

[1] „Verhaltensspezifikation (Pflichtenheft) XCTL-Steuerprogramm“, Version 2.2 von Prof. Klaus Bothe

[2] „Reverse-Engineering der Basisklassen für Dialogfenster“, Version 1.2 von Thomas Kullmann und Günther Reinecker

[3] „Beschreibung einer Schnittstelle zur Motorenansteuerung: Das C-Interface des RTK-Steuerungsprogramms“, Studienarbeit von Sebastian Freund und Derrick Hepp

[4] „semantische Fehler in ursprünglichen Dialogfenster ‚Manuellen Justage’“, Version 1.0 von Thomas Kullmann und Günther Reinecker

[5] „Bewertung der Neuentwürfe des Oberflächenfensters zur Manuellen Justage“, Version 1.7 von Thomas Kullmann und Günther Reinecker

[6] „Layoutkonventionen und Steuerelemente“, Version 1.0 von Thomas Kullmann und Günther Reinecker

IV.2 Index

IV.3 Tabellen

IV.4 Abbildungen