Manual zum C-Quellgenerator IOMSKGEN

Was ist IOMSKGEN?

IOMSKGEN - ***I***nput-***O***utput ***M***a***SK*** ***GEN***erator - ist ein Programm, das C-Quelltext generiert. Es liest Informationen aus einer Beschreibungsdatei und erzeugt daraus ein C-Programmmodul und dessen entsprechende Headerdatei. Die generierten Dateien lassen sich zusammen mit einem in C geschriebenen Anwendungsprogramm compilieren und binden.

Was leistet IOMSKGEN?

Das Tool unterstützt die Programmierung von Anwendungen in der Programmiersprache C, die eine visuelle Kommunikation mit dem Benutzer über einen Bildschirm oder über andere zeichenbasierte Displays erfordern. Es kann sich sowohl um PC-Anwendungen als auch um Bediensoft- bzw. -firmware für embedded Systeme handeln. Die Begriffe Bildschirm und Display werden im folgenden gleichermaßen für das visuelle Ausgabegerät des Systems verwendet.

In der Programmierpraxis entfällt ein großer Anteil der Programmentwicklung auf die Gestaltung der Benutzerschnittstelle. Dies betrifft die Ausgabe von Texten und formatierten Daten auf dem jeweiligen Sichtgerät und die Benutzereingabe über ein Eingabegerät.

IOMSKGEN übersetzt die Anweisungen aus einer Bildschirmobjektbeschreibung, einer sog. ***S***creen-***O***bject-***D***escription, ***.SOD***-Datei, in einen entsprechenden C-Quelltext. Die Screen-object-description ist eine ASCII-Datei und enthält instruktive Anweisungen zum Inhalt, zur Position und Erscheinung und zur Art der Wechselwirkung von Bildschirmobjekten mit dem Anwender.

Ein Entwicklungszyklus mit Hilfe von IOMSKGEN besteht aus der Formulierung bzw. der Änderung der Einträge in der Screen-object-description, wodurch die spätere Gestalt der Anzeige bestimmt wird. Im zweiten Schritt wird die Beschreibung mittels IOMSKGEN übersetzt und schließlich wird die erzeugte Datei als Mitglied des Projekts compiliert und zur Binärdatei gelinkt.

Vorteile des C-Quellgenerators

IOMSKGEN verkürzt die Entwicklungszeit für eine Anwendung erheblich und führt durch sein modulares Konzept zu einer Steigerung der Übersichtlichkeit innerhalb des Projekts.

IOMSKGEN generiert ein in sich konsistentes C-Modul inklusive einer Headerdatei, welches das Display exklusiv verwaltet, d. h. alle Ausgaben des Anwendungsprogramms erfolgen ausschließlich durch das Modul. Zu diesem Zweck exportiert das Modul leistungsfähige C-Funktionen an die Anwendung bzw. den Programmierer.

Damit erfolgt eine klare programmiertechnische Trennung der visuellen Kommunikation von der übrigen Anwendung.

Der Programmierer gestaltet das Aussehen des Displays, die Art der Darstellung der Daten und deren Formatierung, ohne eine Zeile dafür programmieren zu müssen. Die Gestaltung erfolgt weitgehend unabhängig vom Quelltext der eigentlichen Anwendung.

Die Generierung des Quelltextes durch IOMSKGEN erfolgt dynamisch und hängt qualitativ und quantitativ vom Inhalt und Umfang der Screen-object-description ab. Der erzeugte C-Code ist auf einen geringen Verbrauch von Ressourcen und hohe Geschwindigkeit optimiert und entspricht vollständig dem ANSI-Standard. Er wird von nahezu allen C-Compilern reibungslos übersetzt.

IOMSKGEN ist multiscreenfähig, d. h. der generierte Code unterstützt mehrere Bildschirme. Er führt komplexe Prozeduren aus wie Bildschirmwechsel mit Bildaufbau, die Bewegung eines Objektcursors auf dem Display, die formatierte Darstellung der Werte von Variablen und das Editieren unterschiedlicher Variablentypen zur Übergabe an das Anwendungsprogramm. Das Datenformat der Variablen des Anwendungsprogramms wird automatisch berücksichtigt, die Darstellung der Werte wird typkonsistent durchgeführt.

Wird im Rahmen der Produktpflege eine Modifikationen der Gestalt der Anzeige z. B. zur Anpassung an unterschiedliche Benutzergruppen notwendig, so erfolgt die Änderung über die Screen-object-description ohne einen Eingriff in den Quelltext. Es ist lediglich ein IOMSKGEN-MAKE-Zyklus zur Erzeugung des geänderten Programms erforderlich. Dies entspricht einem zeitlichen Vorteil und einer Fehlerreduktion.

Voraussetzungen für die Einbindung des generierten Codes

Ein mittels IOMSKGEN generiertes C-Modul ist dazu bestimmt, die gesamte Verwaltung des Displays zu übernehmen. Zur Kommunikation mit einem Anwendungsprogramm bedarf es einiger weniger Basisroutinen, die der Programmierer dem Modul zur Verfügung stellen muß. Diese Routinen beziehen sich auf die Ansprache der Displayhardware.

Das Display wird als ein zeichenorientiertes Gerät vorausgesetzt. Die folgenden Funktionen müssen zur Ansprache des Displays realisierbar sein:

1. Funktion zur Darstellung eines Zeichens an der momentanen Cursorposition,
2. Funktion zur Positionierung des Cursors,
3. Funktion zur Einschaltung von aufgehellter oder inverser Zeichendarstellung,
4. Funktion zur Einschaltung der normalen Darstellung von Zeichen.

   Besitzt das Ausgabegerät akustische Fähigkeiten, so unterstützt der generierte C-Code
5. eine Funktion zur Erzeugung eines Signaltons.

Während die Funktionen unter 1. bis 4. essentiell für eine Einbindung sind, ist die Funktion unter 5. nicht zwingend erforderlich. Die Unterstützung von Signaltönen dient lediglich dem Editierprozeß im Falle von unzulässigen Eingaben des Anwenders.

Die o. b. Funktionen werden über die erzeugte Headerdatei in das Modul importiert. Dies erfolgt durch Anpassung der in der Datei enthaltenen Funktionsmakros durch den Programmierer. Dieselbe Headerdatei exportiert alle Funktionen des C-Moduls zur Ansprache komplexer Displayoperationen durch Externals an das Anwenderprogramm.

IOMSKGEN stellt einen Zeileneditor zur Verfügung für die interaktive Übergabe von Werten an Variable des Anwendungsprogramms. Die vom Anwender eingegebenen Zeichen erfordern eine Routine, die es erlaubt, Zeichen von der Eingabehardware zu holen. Die Hauptschleife des Anwendungsprogramms muß die eingegebenen Zeichen an das Modul übergeben.

Der Begriff des Bildschirmobjekts

Ein Bildschirmobjekt, screen object, ist ein auf einem Sichtgerät darstellbarer Inhalt. Es wird durch einen Eintrag in der Screen-object-description vollständig charakterisiert. Screen objects sind durch Typen klassifiziert, denen Schlüsselwörter in der SOD-Datei entsprechen. Die folgende Tabelle enthält die verwendbaren Schlüsselwörter und deren Bedeutung.

Screen-Object-Type	Schlüsselwort	Beschreibung
Upper Left Corner	UPLC	obere linke Ecke des Displays
Lower Right Corner	LORC	untere rechte Ecke des Displays
Constant String	CSTR	konstante Zeichenkette
String	STR	variable Zeichenkette im Anwendungsprogramm
Integer variable	INT	Integervariable des Anwendungsprogramms
Float variable	FLOAT	Floatvariable des Anwendungsprogramms
Time	TIME	formatierte Uhrzeit
Date	DATE	formatiertes Datum

Jedes Screen object wird neben seinem Typ durch seine Y,X-Position auf dem Display beschrieben. Die Y-Koordinate entspricht dabei der Zeilennummer, die X-Koordinate der Spaltennummer des ersten dargestellten Zeichens des Objektes. Der erste numerische Eintrag in der SOD-Datei wird als Y-Koordinate, der zweite numerische Eintrag als die X-Koordinate des Objekts interpretiert. Weitere folgende numerische Einträge haben abhängig vom Objekttyp unterschiedliche Bedeutungen.

<KEYWORD>_<Y>_<X>_ ...

Beschreibung der Objekte in der SOD

Die Objekttypen UPLC und LORC

Die Schlüsselwörter *UPLC*, **UP**per **L**eft **C**orner, und *LORC*, **LO**wer **R**ight **C**orner, bezeichnen mit jeweils einem YX-Koordinatenpaar den Punkt der oberen linken Ecke bzw. der unteren rechten Ecke des Displays.

Beispiel:

Die beiden Einträge

UPLC 0 0
LORC 24 79

beschreiben einen Standardbildschirm mit 25 Zeilen und 80 Spalten, wobei die erste Zeile und Spalte jeweils bei 0 beginnt. Die Spezifikation der beiden Objekte ist zwingend. Das generierte C-Modul validiert Befehle der Cursorpositionierung, indem es die Darstellungsgrenzen des Displays prüft. Mit der Deklaration der *UPLC*- und *LORC*-Objekte übergibt der Programmierer die Kontrolle vollständig an das erzeugte C-Modul. Die Objekte *UPLC* und *LORC* besitzen keinen Bezug zum Anwendungsprogramm.

Der Objekttyp CSTR

Der Typ *CSTR* wird verwendet, wenn es um eine Darstellung konstanten Inhaltes geht, z. B. eine Überschrift eines Menüs oder die Bezeichnung einer Beobachtungsgröße wie ‚Temperatur' oder ‚Dichte'.

Beispiel:

Der Eintrag

0 32 CSTR ”Mainmenue”

erzeugt an der YX-Position 0, 32 den Schriftzug ‚Mainmenue'.

Mit der Deklaration eines *CSTR*-Objekts geht die Kontrolle vollständig an das erzeugte C-Modul über. Ein Objekt des Typs *CSTR* besitzt keinen Bezug zum Anwendungsprogramm.

Der Objekttyp INT

Der Objekttyp *INT* wird verwendet, um den Wert einer Integervariablen des Anwendungsprogramms auf dem Display formatiert darzustellen, bzw. es dem Benutzer zu ermöglichen, ihren Wert interaktiv über ein Eingabegerät zu manipulieren und an die Variable zu übergeben.

Beispiel:

Der Eintrag

```
10  2  INT  4  %sample%
```

stellt den Wert der Anwendervariablen ‚sample' an der Position 10, 2 mit max. 4 Dezimalstellen dar. Der Variablenname des Anwendungsprogramms wird in der Notation durch %-Zeichen eingeschlossen.

Mit dem Objekttyp *INT* wird eine Variable des Anwendungsprogramms assoziiert, d. h. die Variable ‚sample' muß in der Applikation durch *int sample;* deklariert sein.

Der Objekttyp FLOAT

Der Objekttyp *FLOAT* wird verwendet, um den Wert einer Floating-point-Variablen des Anwendungsprogramms formatiert darzustellen, bzw. es dem Benutzer zu ermöglichen, ihren Wert interaktiv über ein Eingabegerät zu verändern und an die Variable zu übertragen. Neben den YX-Koordinaten werden als 3. numerischer Parameter die Anzahl der dargestellten Ziffern inkl. Dezimalpunkt und als 4. numerischer Parameter die Anzahl der Nachkommastellen übergeben.

Beispiel:

Der Eintrag

```
11  1  FLOAT  7  2  %temp%
```

stellt den Wert der Anwendervariablen ‚temp' an der Position 11, 1 mit 7 Stellen und 2 Nachkommastellen dar. Wird der 4. Parameter weggelassen, so erfolgt die Darstellung defaultmäßig mit 2 Nachkommastellen. Auch in diesem Fall wird der Variablenname zwischen %-Zeichen notiert.

Mit dem Objekttyp *FLOAT* wird eine Variable des Anwendungsprogramms assoziiert, d. h. die Variable ‚temp' muß in der Applikation durch *float temp;* deklariert sein.

Der Objekttyp STR

Der Objekttyp *STR* wird verwendet, wenn eine Zeichenkette mit veränderlichem Inhalt auf dem Display dargestellt werden soll.

Beispiel:

Der Eintrag

```
23  2 STR  15 %status%
```

stellt einen String an der Position 23, 2 mit einer Länge von maximal 15 Zeichen dar. Sein Inhalt wird durch die Anwendung bestimmt und kann sich im Laufe des Programms ändern.

Mit dem Objekttyp *STR* wird eine Stringvariable des Anwendungsprogramms assoziiert, d. h. die Variable ‚status' muß in der Applikation durch *char *status;* deklariert sein.

Die Objekttypen TIME und DATE

Der Objekttyp *TIME* trägt wie der Typ *DATE* dem Bedürfnis des Anwenders Rechnung, das aktuelle Datum und die Uhrzeit auf einem Display zur Verfügung zu haben. Beide Objekte sind komplexe Screen objects und benötigen einen Bezug zur Realtime clock des Systems.

Die Uhrzeit wird im festen Format

HH:MM:SS

angezeigt.

IOMSKGEN bietet beim Format der Datumsanzeige **N**ational **L**anguage **S**upport NLS und unterstützt derzeitig 4 Sprachen. Die Sprache wird zusammen mit dem Schlüsselwort *DATE* als zweibuchstabiger NLS token in der Screen-object-description übergeben.

Sprache	**NLS token**	**Datumsformat**
Deutsch	”GE”	So xx Okt 1999
Englisch	”UK”	Sun xx oct 1999
Französisch	”FR”	dim xx oct 1999
Spanisch	”ES”	dom de xx oct de 1999

Beispiel:

Die beiden Einträge

```
24  60 DATE  ”FR”
24  72 TIME
```

bewirken die Darstellung des Datums an der Position 24, 60 in französischer Sprache und der Uhrzeit an der Position 24, 72.

Beschreibung der Funktionen von IOMSKGEN

Im folgenden wird das API zum generierten C-Quellcode beschrieben. Es handelt sich um eine Gruppe von C-Funktionen und Funktionsmakros, durch welche das Modul mit einer Anwendung kommuniziert.

Das Konzept von IOMSKGEN sieht eine exklusive Verwaltung des Displays durch das generierte C-Modul vor. Dennoch stellt das Modul dem Anwender neben den komplexen Screen object-bezogenen Routinen primäre Ausgabefunktionen zur Verfügung.

clear()	refresh()	move()	addch()	addstr()	mvaddch()
mvaddstr()	getyx()	select_scr()	step_2_obj()	move_2_obj()	put_param()
get_param()	access_param()	edit_param()			

Die Funktion *move()*

Deklaration:	int move(int row, int column);
Funktion:	*move()* bewegt den Cursor an die übergebene YX-Position auf dem Display.
Input:	YX-Koordinaten ‘row’, ‘column’
Return:	OK, wenn Positionierung des Cursors möglich, ERR, wenn außerhalb der Grenzen des Displays

Die Funktion *addch()*

Deklaration:	void addch(int c);
Funktion:	addch() schreibt ein Zeichen an der momentanen Cursorposition in den Bildschirm. Die Funktion bewegt den Cursor automatisch um eine Stelle nach rechts. addch() führt die aktuelle Cursorposition ‘cury’ und ‘curx’ mit.
Input:	Zeichen ‘c’
Return:	none
⇒ Anwendung:	*addch()* erlaubt die Ausgabe eines Zeichens, ohne daß dies zuvor in der Screen-object-description vereinbart zu sein braucht. Es besteht jedoch selten Bedarf, einzelne Zeichen in das Display zu schreiben.

Die Funktion *addstr()*

Deklaration:	void addstr(char *s);
Funktion:	Mittels *addstr()* wird eine Zeichenkette an der momentanen Cursorposition auf dem Display ausgegeben.
Input:	Zeichenkette ‘s’
Return:	none
⇒ Anwendung:	*addstr()* erlaubt die Ausgabe einer Zeichenkette, ohne daß diese zuvor in der Screen-object-description vereinbart zu sein braucht.

Das Funktionsmakro *mvaddch()*

Deklaration:	void mvaddch(int row, int column, int c);
Funktion:	Die Funktion schreibt das übergebene Zeichen an der übergebenen Cursorposition.
Input:	YX-Koordinaten ‘row’, ‘column’, Zeichen ‘c’
Return:	none
⇒ Anwendung:	*mvaddch()* ist eine kompakte Form der positionierten Ausgabe einzelner Zeichen.

Das Funktionsmakro *mvaddstr()*

Deklaration:	void mvaddstr(int row, int column, char *s);
Funktion:	*mvaddstr()* schreibt eine Zeichenkette an der übergebenen Cursorposition.
Input:	YX-Koordinaten ‘row’, ‘column’, Zeichenkette ‘s’.
Return:	none
⇒ Anwendung:	*mvaddstr()* ist eine kompakte Form der positionierten Ausgabe von Zeichenketten.

Das Funktionsmakro *getyx()*

Deklaration:	void getyx(int row, int column);
Funktion:	*getyx()* gibt die aktuelle Cursorposition zurück.
Return:	none

Die Funktion *clear()*

Deklaration:	void clear(void);
Funktion:	*clear()* loescht den ausgewählten Bildschirm.
Input:	none
Return:	none

Die Funktion *refresh()*

Deklaration:	void refresh(void);
Funktion:	*refresh()* baut den gesamten Bildschirminhalt auf, d. h. alle screen-objects werden mit ihren ggf. assoziierten Variablenwerten dargestellt.
Input:	none
Return:	none
⇒ Anwendung:	*refresh()* wird zum grundsätzlichen Aufbau eines Bildschirms oder zur Restaurierung des Bildschirminhaltes aufgerufen.

Die Funktion *select_scr()*

Deklaration:	int select_scr(char scr_indx);
Funktion:	Die Funktion schaltet auf den Bildschirm mit dem übergebenen Index um. Die Numerierung beginnt bei 0, d. h. der erste definierte Bildschirm hat den Index 0.
Input:	Index des Bildschirms ‘scr_indx’
Return:	OK, wenn der übergebene Bildschirmindex gültig ist, ERR, wenn zum übergebenen Index kein Bildschirm existiert, oder wenn auf ein Screen object des aktuellen Bildschirms zugegriffen (editiert) wird.

Die Funktion *step_2_obj()*

Deklaration:	int step_2_obj(char to);
Funktion:	Wird die Funktion mit dem Wert _NEXT aufgerufen, so bewegt sie den Objektcursor zum nächsten screen-object, ein Aufruf mit _PREV führt zu einer Bewegung des Objektcursors zum vorangehenden screen-object. Die Anordnung der Screen objects entspricht der Reihenfolge der Einträge in der screen-object-description.
Input:	Bewegungsrichtung ‘to’ aus [_NEXT, _PREV]
Return:	
⇒ Anwendung:	*step_2_obj* unterstützt die intuitive Navigation des Anwenders auf dem Display. Durch entsprechende Übersetzung z. B. der ”Pfeiltasten” und Aufruf von *step_2_obj()* kann sich ein Benutzer zwischen den Screen objects bewegen.

Die Funktion *move_2_obj()*

Deklaration:	int move_2_obj(void *adr);
Funktion:	Die Funktion bewegt den Objektcursor zu jenem screen-object, mit welchem die übergebene Variable assoziiert ist.
Input:	Adresse der Variablen ‘adr’
Return:	OK, wenn im aktuellen Bildschirm ein screen-object gefunden wurde, das mit der übergebenen Variablen assoziiert ist, ERR, wenn kein assoziiertes Screen object im aktuellen Bildschirm gefunden wurde.
⇒ Anwendung:	Im Gegensatz zur Funktion *step_2_obj()* erlaubt *move_2_obj()* die gezielte Bewegung des Objektcursors zur gewünschten assoziierten Variablen. Die Aktion wird weniger durch Benutzerintervention wie im Falle von *step_2_obj()* ausgelöst, sondern unterstützt dogmatische Zielvorgaben des Anwendungsprogramms.

Die Funktion *put_param()*

Deklaration:	int put_param(void *adr);
Funktion:	*put_param()* stellt den Wert einer übergebenen Variablen auf dem Display dar. Die Funktion arbeitet unabhängig davon, wo sich der Objektcursor gerade befindet.
Input:	Adresse der Variablen ‘adr’
Return:	OK, wenn die übergebene Variable tatsächlich mit einem screen-object des ausgewählten Bildschirms assoziiert ist, ERR, wenn die Variable mit keinem der Mitglieder der screen-objects des aktuellen Bildschirms assoziiert ist.
⇒ Anwendung:	Die Funktion transportiert die Werte von Anwendungsvariablen zur Beobachtung in das Display.

Die Funktion *get_param()*

Deklaration:	void *get_param(void);
Funktion:	Die Funktion gibt die Adresse der mit dem fokussierten Screen object assoziierten Variablen zurück.
Input:	none
Return:	Adresse der assoziierten Variablen.

⇒ Anwendung: *get_param()* ermöglicht dem Anwendungsprogramm die Zuordnung zwischen dem gerade fokussierten Screen object und seiner assoziierten Variablen.

Die Funktion *access_param()*

Deklaration: int access_param(void);
Funktion: Die Funktion greift folgendermaßen auf ein Screen object zu:
Sie identifiziert den Typ der assoziierten Variablen und wandelt deren aktuellen
Wert in einen String um, welcher auf einen Zeichenpuffer kopiert wird. Schließlich öffnet sie den Zeileneditor, so daß der Pufferinhalt vom Anwender mittels der Funktion edit_param() modifiziert werden kann.
Input: none
Return: OK, wenn ein Zugriff auf die assoziierte Variable möglich ist. Der Editor wird in diesem Fall geöffnet.
ERR, wenn keine assoziierte Variable gefunden werden konnte, bzw. wenn bereits auf ein Screen object zugegriffen wird.

Die Funktion *edit_param()*

Deklaration: int edit_param(int c);
Funktion: Die Funktion verarbeitet das übergebene Zeichen in der Art eines Zeileneditors. Es muß zuvor auf ein screen-object mittels der Funktion access_param() zugegriffen worden sein. Das Zeichen wird nicht dem Typ der assoziierten Variablen entsprechend interpretiert, sondern es wird als Element einer Zeichenkette behandelt. Eine Interpretation der übergebenen Zeichen findet am Ende der gesamten Eingabe statt.
Input: Zeichen ‘c‘
Return:
OK, wenn das übergebene Zeichen verarbeitet wird, d. h. der Editor ist aktiv und das Zeichen wird entsprechend auf dem Display dargestellt.
ERR, wenn auf kein screen-object zugegriffen wurde, *edit_param()* verhält sich in diesem Fall passiv.
_READY, nach Übergabe von CR oder LF, wenn eine Eingabe erfolgreich abgeschlossen wurde, d. h. wenn die Syntaxprüfung fehlerfrei war.
⇒ Anwendung: *edit_param()* ermöglicht die interaktive Benutzereingabe von numerischen und textuellen Größen.

Beschreibung des Zeileneditors

IOMSKGEN stellt einen Zeileneditor zur Verfügung als ein Instrument zur interaktiven Kommunikation mit dem Anwender. Der Editor ermöglicht die Eingabe von numerischen Werten und Texten im laufenden Anwendungsprogramm. Die folgenden Aufgaben werden vom Editor wahrgenommen:

1. Zeichen, welche der Anwender über ein Eingabegerät an die Anwendung schickt, werden auf dem Bildschirm dargestellt. Gleichzeitig wird der Inhalt eines Zeilenpuffers entsprechend mit der Eingabe synchronisiert..

2. Die eingegebenen Zeichen werden hinsichtlich ihrer Darstellbarkeit gefiltert und der Zeichenpuffer auf Überlauf überwacht.

3. Der Editor unterstützt Cursorbewegungen, Sprünge auf den Zeilenanfang bzw. das Zeilenende, das Löschen eines Zeichens unter dem Cursor und rückwärtiges Löschen von Zeichen.

4. Ist ein akustisches Ausgabegerät vorhanden und wurde dieses vom Programmierer entsprechend eingebunden, so wird auf Fehleingaben des Anwenders akustisch aufmerksam gemacht.

5. Der Editiermodus wird standardmäßig durch das CR-Zeichen, #13, oder durch einen Zeilenvorschub LF, #10, konstruktiv beendet. Das Steuerzeichen ESC, #27, führt zum Abbruch des Editierens.

6. Nach Abschluß der Eingabe erfolgt eine typbezogene Prüfung der Syntax der eingegebenen Zeichenkette.

Beschreibung eines Editierzyklus

Die Zeicheneingabe des Benutzers erfolgt transparent. Damit ist gemeint, daß alle laufenden Anzeigeprozesse, z. B. von Beobachtungsgrößen, während der Eingabe weiter laufen.

Das Eingabeinterface

Das interaktive Editieren setzt ein Eingabegerät voraus, das über die folgende minimale Schnittstelle ansprechbar ist:

Es existiert eine Funktion, die Zeichen von der Eingabehardware abholt. Diese Funktion möge getch() heißen. getch() muß die Kontrolle umgehend an das aufrufende Programm zurückgeben, auch wenn kein Zeichen bereit steht. Der Zustand ”kein Zeichen bereit” wird dabei durch den Returnwert EOF = 0 signalisiert. Durch eine Normierung der Steuercodes kann jede Eingabehardware in den Editierprozeß eingebunden werden.

Der Editor arbeitet mit der Funktion access_param() zusammen.

Der Zugriff des Anwenders auf eine Variable erfolgt durch die Bewegung des Objektcursors zum Screen object, mit welchem die Variable assoziiert ist. In der Anwendung wird ein Eingabezeichen vereinbart, welches den Zugriff auf das Screen object auslöst und zwar durch den Aufruf von access_param(). Der Zugriff wird also prinzipiell durch die Anwendung autorisiert und erfolgt nicht a priori.

Die Funktion access_param() benötigt keinen Parameter. Sie identifiziert den Typ der assoziierten Variablen selbständig und wandelt deren aktuellen Wert in einen String um, der anschließend auf einem internen Zeichenpuffer zum Editieren bereitsteht. Schließlich öffnet die Funktion den Zeileneditor, d. h. die Funktion edit_param() verarbeitet von nun an Zeichen. edit_param() ist selektiv aktiv. Dies bedeutet, daß die Funktion generell innerhalb der Hauptschleife des Anwendungsprogramms zyklisch wie eine Task aufgerufen werden kann. Sie verhält sich passiv im Fall, daß auf kein Screen object zugegriffen wird.

Nach Öffnen des Editors werden alle vom Anwender eingegebenen Zeichen durch getch() abgeholt und an die Funktion edit_param() übergeben. edit_param() verarbeitet die Zeichen entsprechend der o. a. Beschreibung der Funktion. Der Editierprozeß ist im oben beschriebenen Sinne transparent. Eine Übergabe der Zeichen CR oder LF leitet den Abschluß einer Eingabe ein. Es erfolgt eine

Formatprüfung entsprechend dem Typ der Variablen. Ist die Eingabe fehlerhaft, so kehrt das Modul zum Editierprozeß zurück und gibt dem Benutzer Gelegenheit, seine Eingabe zu korrigieren.

Nach korrekter typbezogener Eingabe erfordern Anwendungsprogramme häufig eine Prüfung, ob der vom Benutzer eingegebene Wert einer Variablen innerhalb eines zulässigen Intervalls liegt. Diese Prüfung sollte nach Abschluß des Editiervorganges automatisch erfolgen. Erst nach erfolgreicher Intervallprüfung wird der neue Wert an die Variable übergeben.

Hard- & Software
Projektierung u. Entwicklung
Robert Warnke

Tel.: 0228 67 72 87
Fax: 0228 68 96 347
Mobil: 0163 78 53635 (Hotline)
E-Mail: robert.warnke@t-online.de