MACRO - Interpreter

picCOLOR

Wissenschaftliche Bildverarbeitung und Bildanalyse

Forschungsinstitut für Bildverarbeitung, Umwelttechnik und Strömungsmechanik
Dr. R. H. G. Müller, Beim Elbkurhaus 12, D -22587 Hamburg
Tel.: (49) - 40 - 24421280 FAX: (49) - 40 - 24421282
email: info@fibus.org WWW: http://www.fibus.org

Hinweise zur picCOLOR Software
Diese Software-Beschreibung zum picCOLOR Bildverarbeitungssystem ist mit großer Sorgfalt geschrieben und hergestellt worden. Trotzdem kann natürlich keine Garantie für Fehlerfreiheit gegeben werden. Technische Änderungen und Verbesserungen der Software sind ohne Ankündigung vorbehalten. Änderungen der Software, die noch nicht in diese Version des Handbuches aufgenommen werden konnten, oder Abweichungen der Software vom Handbuch sind im aktuellen README-File auf den Installationsdisketten aufgeführt. Bitte lesen Sie die dort gegebene Information sorgfältig. Über Anregungen und Verbesserungswünsche, die sowohl die Software als auch die Software-Beschreibung betreffen, ist der Autor jederzeit dankbar. Anregungen werden, soweit technisch und mit vertretbarem Aufwand möglich, in die nachfolgenden Versionen einfließen.

Einzelbenutzer Lizenzabkommen
Die im Software-Paket eingeschlossene Lizenzvereinbarung regelt die zulässige Benutzung von Software und dem vorliegendem Handbuch. Jede unauthorisierte Vervielfältigung der Software oder des Handbuches - auch in Auszügen - ist verboten und verletzt das Lizenzabkommen.

Copyright
Diese Dokumentation ist urheberrechtlich geschützt. Alle Rechte, auch der Übersetzung, des Nachdrucks oder der Vervielfältigung des Handbuches, oder Teilen daraus, vorbehalten. Kein Teil des Handbuches darf ohne schriftliche Genehmigung vom Author - Dr. Reinert H. G. Müller - in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren), auch nicht für Zwecke der Unterrichtsgestaltung, reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Garantie- und Haftungsausschluß
Das picCOLOR Programm und die hier vorliegende Programmbeschreibung sind nach bestem Wissen erstellt und mit Sorgfalt getestet worden. Dennoch sind Fehler nicht völlig auszuschließen. Insbesondere kann nicht garantiert werden, daß die Funktion des picCOLOR Programmes in allen Punkten exakt mit der Beschreibung in diesem Handbuch übereinstimmt. Im Handbuch sind einige Funktionen beschrieben, die im Programm nicht oder noch nicht aufgenommen sind. Die Anwendung des Programmes und der daraus erhaltenen Ergebnisse und Daten unterliegt allein der Verantwortung des Benutzers. Autor und Vertriebspartner können keine Garantie, keine juristische Verantwortung und keine Haftung für Folgen übernehmen, die auf fehlerhafte Angaben im Programm picCOLOR oder im Handbuch zurückzuführen sind.

THE PICCOLOR PROGRAM IS NOT FAULT TOLERANT AND IS NOT DESIGNED, MANUFACTURED, OR INTENDED FOR USE OR RESALE AS ONLINE CONTROL EQUIPMENT IN HAZARDOUS ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, SUCH AS IN THE OPERATION OF
NUCLEAR FACILITIES, AIRCRAFT NAVIGATION OR COMMUNICATION SYSTEMS, AIR TRAFFIC CONTROL, DIRECT LIFE SUPPORT MACHINES, OR WEAPONS SYSTEMS, IN WHICH THE FAILURE COULD LEAD DIRECTLY TO DEATH, PERSONAL INJURY, OR SEVERE PHYSICAL OR ENVIRONMENTAL DAMAGE.

Geschützte Bezeichnungen
Alle im Handbuch benutzten Bezeichnungen, die als Warenzeichen oder eingetragene Warenzeichen bekannt sind, sind im folgenden aufgelistet. Die Vollständigkeit dieser Liste kann jedoch nicht garantiert werden. Die Benutzung eines hier nicht erfaßten Warenzeichens darf nicht als Mißachtung der Gültigkeit dieses Warenzeichens angesehen werden.
Texas Instruments ist eingetragenes Warenzeichen der Texas Instruments Corporation. IBM, Personal Computer-AT, PC/AT sind eingetragene Warenzeichen und PC-DOS, VGA sind Warenzeichen von International Business Machines Corporation. Microsoft ist eingetragenes Warenzeichen und Windows, WINDOWS NT, Internet Explorer und MS-DOS sind Warenzeichen der Microsoft Corporation. Adobe ist Warenzeichen der Adobe Corporation. Netscape ist eingetragenes Warenzeichen der Netscape Corporation. HP und PCL sind eingetragene Warenzeichen der Hewlett Packard Company.

Inhalt

Macro-Interpreter für picCOLOR 5

-Kommandointerpreter.....5

-Funktionsinterpreter.....8

Funktionsbeispiel: TOP-HAT.....13

Funktionsbeispiel: Bildübernahme.....14

Macro-Interpreterstruktur und Syntax..... 15

Macro-Funktionstabellenstruktur.....16

Inhaltsverzeichnis der MACRO-Funktionen.....17

Funktionen und Parameter.....18

Anhang: Fehlermeldungen.....130

Macro-Interpreter für picCOLOR

Das picCOLOR Programmsystem besitzt einen sehr leistungsfähigen Macro-Interpreter. Mit Hilfe dieses Interpreters können Bildbearbeitungsvorgänge, die aus mehreren verschiedenen Bearbeitungsschritten bestehen, völlig automatisch oder halbautomatisch, d. h. mit interaktiver Eingriffsmöglichkeit, ablaufen. Der Macro-Interpreter besteht aus einem Kommandointerpreter und einem Funktionsinterpreter. Die Kommandosprache setzt sich aus verschiedenen Strukturen zur Steuerung des Programmablaufes sowie aus Variablen-Zuweisungen zusammen. Der Funktionsinterpreter kann praktisch alle im picCOLOR Programm vorhandenen Funktionen ausführen. Es wird hierbei zwischen interaktiven Funktionen und sogenannten ‘‘Batch’’-Funktionen, d. h. automatisch ohne Benutzereingriff ablaufenden Funktionen unterschieden. Einige Funktionen liegen in beiden Ausführungen vor: so kann z. B. zum Laden eines Bildes die Load-Dialogbox auf den Bildschirm gebracht werden, es kann jedoch auch direkt ein Lade-Kommando mit festem oder variablem Filenamen ausgeführt werden. Mit der Macrofunktion können nur Funktionen und Bearbeitungen ausgeführt werden, die im picCOLOR Programm (incl. den jeweils eingebundenen Zusatz-Modulen und dem USER-Modul) programmiert sind. Es können keine neuen Funktionen definiert werden, außer es sind aus vorhandenen Funktionen zusammengesetzte Funktionsabläufe. Wenn völlig neue Funktionen benötigt werden, z. B. ein spezieller Konvolutionsfilter, so muß dieser mit dem USER-Modul erstellt werden.

Der Kommando-Interpreter

Im folgenden soll eine kurze Beschreibung des Kommando-Interpreters gegeben werden, mit deren Hilfe es leicht gelingen sollte, sich in die picCOLOR-Macro-Sprache einzuarbeiten. Eine exakte Beschreibung der Sprachstruktur (Syntax der MACRO-Sprache in der Backus-Naur Form (BNF)) wird weiter unten aufgeführt.

Reservierte Namen

DEFINT, DEFFLT, IVAR, FVAR, STRING, FOR, ENDFOR, WHILE, ENDWHILE, IF, ELSE, ENDIF, CONCAT, LTOA, ITOA, FTOA, SKIP, STOP, CALL, FUNCTION, RETURN, STRCMP, GOTO, LABEL

Anhand der reservierten Namen kann schon abgeschätzt werden, welche Kontrollstrukturen programmiert werden können. Diese reservierten Namen müssen immer groß geschrieben werden.

Beim **FOR-Loop** wird vor Ausführung des Loops festgelegt, wie oft der Loop durchlaufen wird. Dies wird durch das Ergebnis der ‘‘expression’’ angegeben. ‘‘expression’’ kann eine Integer-Konstante oder eine Integer-Variable sein. Ein FOR-Loop kann auch 0-mal, also keinmal durchlaufen werden. Bei negativer ‘‘expression’’ erfolgt ein Fehlerabbruch.

```
FOR expression
  command
  ...
ENDFOR
```

Folgendes Beispiel führt 5-mal den Bildschärfe-Operator laplace_3x3 mit dem Parameter (4) aus:

```
FOR 5
  laplace_3x3  ( 4 )
ENDFOR
```

Statt der ‘‘5’’ könnte auch eine Variable stehen:

```
IVAR[1] := 5
FOR IVAR[1]
  laplace_3x3 ( 4 )
ENDFOR
```

Die Klammerung der Parameter ist nicht unbedingt notwendig und dient nur zur besseren Lesbarkeit. Der Wert der Loop-Variable wird nicht verändert.

Beim **WHILE-Loop** wird während der Ausführung bestimmt, wie oft der Loop durchlaufen wird. Hierzu wird vor jeder Ausführung die ‘‘condition’’ überprüft. Die ‘‘condition’’ besteht aus zwei ‘‘expressions’’, die über eine konditionale Operation verbunden sind. ‘‘expression’’ kann wieder eine Konstante oder Variable (Integer oder Float) sein, mögliche konditionale Operatoren sind: ‘=’, ‘#’, ‘>’, ‘<‘, ‘>=’, ‘<=’.

```
WHILE  condition
 command
 ...
ENDWHILE
```

Das folgende Beispiel führt den laplace_3x3-Befehl 5-mal aus:

```
IVAR[1] := 10
WHILE  IVAR[1]  >  5
 IVAR[1]--
 laplace_3x3 ( 4 )
ENDWHILE
```

Bei der konditionalen Bedingung (**IF - ELSE - ENDIF**) wird ebenfalls die ‘‘condition’’ getestet. Wenn sie wahr ist, wird der zwischen IF und ELSE liegende Programmteil ausgeführt. Wenn sie unwahr ist, wird der zwischen ELSE und ENDIF liegende Programmteil ausgeführt. Es muß immer ein ELSE-Programmteil vorhanden sein. Dieser kann jedoch auch leer sein oder nur einen SKIP-Befehl enthalten, der nur als Platzhalter zur besseren Lesbarkeit gedacht ist und keine Wirkung hat.

```
IF condition
 command
 ...
ELSE
 command
 ...
ENDIF
```

SKIP ist eine leere Anweisung. Sie bewirkt nichts und ist dafür vorgesehen, z. B. als Platzhalter für leere ELSE-Anweisungen zu stehen. Die Benutzung dient nur der Übersichtlichkeit des Programmes und ist nicht zwingend vorgeschrieben.

Der **STOP**-Befehl beendet das Macro-Programm sofort.

Unterprogramme (Subroutinen) werden mit der **FUNCTION**-Anweisung definiert und über die **CALL**-Anweisung aufgerufen:

```
(Hauptprogramm)
...
command
CALL 1
command
...
STOP

FUNCTION 1
command
...
RETURN
```

Funktionsdefinitionen müssen am Ende des Macro-Programmes stehen - sie müssen durch einen STOP-Befehl vom Hauptprogramm getrennt sein, da sonst das Programm in die Funktion hineinliefe.

Variablen-Zuweisung: Es gibt 3 Arten von Variablen: Integer-Variable, Floating-Point-Variable und String-Variable. Die Integer- und Float-Variablen sind jeweils als Array implementiert. Die derzeitige Array-Größe ist 512 Variable, von denen jedoch nur 256 Variable frei benutzt werden sollten, wie weiter unten erklärt wird. D. h. man kann die Integer-Variablen **IVAR[0]** bis **IVAR[255]** und die Floating-Point-Variablen **FVAR[0]** bis **FVAR[255]** benutzen. Der Index wird dabei in eckige Klammern gesetzt. Selbstverständlich kann der Index auch wieder eine Variable sein: IVAR[IVAR[5]] ist eine gültige ‘‘expression’’. Zusätzlich zu den vordefinierten Array-Variablen können über die Befehle **DEFINT** und **DEFFLT** Variable mit beliebigen Namen definiert werden. Dies sollte am Anfang eines Programmes geschehen, kann aber prinzipiell auch beliebig innerhalb des Programmes getan werden. Alle Variablen sind global. Für jede Variablen-Definition wird eine eigene Programmzeile benötigt:

DEFINT Integername
DEFFLT Floatname

Die Bezeichnungen sind groß/klein-sensitiv. Es werden lediglich die 12 ersten Zeichen eines Namens zur Unterscheidung benutzt. Es wird empfohlen, Anfangsbuchstaben der Variablen zur Unterscheidung von Funktionen groß zu schreiben. Intern werden diese definierten Variablen ebenfalls als Array-Variable, allerdings mit den Indizes [256] bis [512] geführt. Eine Benutzung dieser internen Array-Indizes über die IVAR[n] oder FVAR[n]-Bezeichnugen führt zwar nicht zum Fehlerabbruch, auch wenn die entsprechende Variable mit dem benutzten Index bereits definiert worden ist, sollte jedoch auf keinen Fall geschehen. Wenn also eine Variable Xyz mittels DEFINT Xyz definiert wurde, so ist sie intern z. B. der Variablen IVAR[256] zugewiesen und die fälschliche Benutzung von IVAR[256] führt nicht zum Fehlerabbruch. Hier ist Vorsicht geboten!

Bei gemischter Benutzung von Integer- und Floating-Point-Variablen wird eine Typ-Konvertierung durchgeführt. Dabei wird bei einer Zuweisung auf eine Integer-Variable ein Floating-Point-Wert arithmetisch gerundet. Zuweisungen von Strings auf Integer oder Float-Werte sind erlaubt. Die Umwandlung geschieht automatisch.

Zuweisungen geschehen über den Zuweisungsoperator ‘:=’. Weiterhin gelten die Inkrement- und Dekrement-Operatoren ‘++’ und ‘--’, die, an eine Variable angehängt, dieselbe um die Konstante ‘1’ inkrementieren oder dekrementieren. Duale Operatoren sind ‘+=’, ‘-=’, ‘*=’, ‘/=’, ‘<<=’, ‘>>=’, ‘&=’, ‘|=’, ‘^=’, ‘%=’. Hiermit können die vier Grundrechenarten mit Konstanten oder Variablen durchgeführt werden und, wie in "C" für Integer-Variable definiert, shift-Operationen, arithmetische UND, ODER, EXCLUSIVE-ODER und die modulo-Funktion. Für Floating Point Variable gelten allerdings nur die vier Grundrechenarten ‘+=’, ‘-=’, ‘*=’, ‘/=’.

Beispiel: IVAR[1] += 5 bedeutet, daß auf die Variable IVAR[1] die Konstante ‘5’ aufaddiert wird.
Beispiel: FVAR[1] /= FVAR[2] dividiert die Variable FVAR[1] durch die Variable FVAR[2].

Diese Notation entspricht weitgehend der in der Programmiersprache ‘C’ definierten Notation. Als Einschränkung muß z. Zt. noch gelten, daß auf beiden Seiten der hier genannten Operatoren immer nur einfache ‘‘expressions’’, d. h. Konstanten oder Variable stehen können. Eine komplexere Berechnung muß also z. Zt. aus Einzelschritten in mehreren Zeilen etwas mühsam ‘zusammengebaut’ werden.

Bestimmte Variable sind vordefiniert und werden vom System benutzt: so ist IVAR[0] der Ausführungsrückgabewert von Funktionen (1:ok, 0:warning, -1:error), und IVAR[10]..IVAR[19] sowie FVAR[10]..FVAR[19] werden für die Parameterzurückgabe von Funktionen benutzt. (Siehe auch weiter unten, Funktionsbeschreibung)

Die String-Variablen mit dem Namen **STRING** sind ebenfalls als Array implementiert. In der derzeitigen Implementation sind 16 Strings definiert, STRING[0] bis STRING[15]. Diese Strings sind jeweils wiederum Arrays von maximal 512 Zeichen. Ein String wird über den einfachen Befehl:

STRING[0] := ‘‘Dies ist ein Test-String’’

zugewiesen. Stringfunktionen geben einen String in STRING[1] an das Macro-Programm zurück. Da für den automatischen Programmablauf mit z. B. einer ganzen Reihe von Bildern oft ein Name mit einer laufenden Nummer versehen werden muß, wurden die Funktionen ‘**CONCAT**’, ‘**LTOA**’ (=‘**ITOA**’) sowie ‘**FTOA**’ und ‘**ETOA**’ implementiert. Die letzteren Formate ergeben das FloatingPoint-Format mit 6 Nachkommastellen, wahlweise in der normalen Float- oder der Exponentialdarstellung.

Die String-Concatenation mit der **CONCAT-**Funktion benötigt drei Parameter, den ersten String, den zweiten String und die Gesamtlänge, die der Ergebnis-String haben soll. Wenn Beide Strings zusammen kürzer als die vorgegebene Gesamtlänge sind, wird zwischen beiden Strings mit ‘0’ aufgefüllt, so daß hiermit Filenamen mit laufender Nummer und gleicher Länge zusammengestellt werden können. Wenn die Strings einfach nur aneinandergehängt werden sollen, so muß als Gesamtlänge eine negative Zahl eingegeben werden. Das Ergebnis der Funktion ist wieder ein String. Mit den folgenden Strings

STRING[1] := ‘‘test’’
STRING[2] := ‘‘string’’
erhält man mit dem Befehl:
STRING[0] := CONCAT (STRING[1], STRING[2], 8)
den String ‘‘teststri’’, mit dem Befehl:
STRING[0] := CONCAT (STRING[1], STRING[2], -1)
den String ‘‘teststring’’. Mit den Funktionen **‘LTOA’**, **‘FTOA’** und ‘**ETOA**’ können Integer- oder Floating-Point-Variable in Strings umgewandelt werden. Das Ergebnis beider Funktionen ist jeweils ein String. Die Befehlsfolge

IVAR[1] := 1
STRING[0] := ‘‘test’’
STRING[1] := CONCAT (STRING[0], LTOA(IVAR[1]), 8)
load_filename STRING[1]

erzeugt den String ‘‘test0001’’ und weist ihn dem internen ‘‘Load’’-Filenamen zu.

Die **STRCMP**-Funktion erlaubt einen Vergleich zweier Strings. Der Rückgabewert liegt dann in IVAR[10]. Er gibt das Vergleichsergebnis in "C"-Konvention aus: IVAR[10] = 0: Strings sind gleich; IVAR[10] < 0: string1 ist kleiner als string2; IVAR[10] > 0: string1 ist größer als string2. Größer heißt dabei nicht länger! Der Absolutbetrag des Returnwertes ist der ASCII-Code des ersten ungleichen Zeichens.

Kommentarzeilen sind erlaubt. Sie sind wie in "C"-Konvention mit /* und */ einzurahmen und können sich auch über mehrere Zeilen erstrecken. Zeilen können jedoch auch durch einen Asterisk * am Zeilenbeginn auskommentiert werden. Dieser darf dann jedoch nicht eingerückt sein.

Die Benutzung von **Leerzeilen** sowie das **Einrücken** ist erlaubt. Zum Einrücken können beliebige Delimiter benutzt werden. Delimiter sind die folgenden Zeichen: Blank (Leerzeichen), Tab, Komma. Zusätzlich sind auch noch runde und eckige Klammern als Delimiter definiert. Sie sollten jedoch nur für Array-Bezeichnungen und für Funktionsparameter benutzt werden.

Sprungbefehle und dazugehörende Adresslabels (**GOTO und LABEL**) sind zwar implementiert, sind jedoch nicht für die allgemeine Benutzung freigegeben, da sie die Programmstruktur zerstören. Eine Anwendung führt zu einem Fehlerabbruch. Sie werden nur intern vom Kommandointerpreter benutzt.

Ein Abbruch des MACRO-Programmes ist über die ESC-Taste möglich. Da die ESC-Taste nicht immer, d. h. nicht in allen Funktionen wirksam ist, muß sie eventuell mehrmals gedrückt werden. Leider gibt es auch Fälle, wo in einem Loop sehr viele Ausgaben hervorgerufen werden, die vom Benutzer quittiert werden müssen. Hier funktioniert die ESC-Taste in der Regel kaum. Oft hilft es dann, mit der Maus die Schließfunktion (kleines Kreuz rechts oben) der Message-Box zu klicken und gleichzeitig die ESC-Taste festzuhalten.

Funktionsinterpreter

Der Funktionsinterpreter kennt praktisch alle im picCOLOR Programm oder im USER-Modul aufgeführten Funktionen und deren Art und Anzahl von Parametern. Es gibt 10 verschiedene Arten von Parameterzusammenstellungen. Diese sind nach der folgenden Tabelle mit einer Kennziffer für Typ und Anzahl der Parameter kodiert und werden in der Funktionsliste weiter unten benutzt:

0 : kein Parameter	5: 1 Floating Point
1 : 1 Integer	6: 2 Floating Point
2 : 2 Integer	7: 1 Integer, 1 Floating Point
3 : 3 Integer	8: 2 Integer, 2 Floating Point
4 : 4 Integer	9: 1 Character String
	10: komplexere Parameterstruktur (nicht implementiert)

Diese Kodierung wird nur intern zur Abfrage und zur Überprüfung der Parameter benutzt. Als Anwender brauchen Sie diese auch in der Funktionsliste angegebenen Ziffern nicht zu berücksichtigen.

Alle Funktionsnamen von picCOLOR werden klein geschrieben. Ein Funktionsaufruf sieht dann folgendermaßen aus:

funktionsname (Parameter1, Parameter2, ...)

1.Beispiel: Es soll die Konstante ‘50’ auf das aktuelle Bild aufaddiert werden. Die hierzu benutzte Funktion ist die "const_op"-Funktion, die in der Funktionsliste wie folgt defiiniert ist:

```
const_op        8, B        type, iconst: integer; fconst, dummy: floating point;
```

Der dazugehörende Befehl sieht folgendermaßen aus (type=2 ist die Nummer der "add"-Funktion, die Additionskonstante ist iconst=50, und die Float-Parameter fconst=0.0 und dummy=0.0 werden nicht verwendet):

```
const_op ( 2, 50, 0.0, 0.0 )
```

oder mit Benutzung einer Variablen:

```
IVAR[1] := 50
const_op ( 2, IVAR[1], 0.0, 0.0 )
```

Dabei können die runden Klammern weggelassen werden und durch andere Delimiter, z. B. durch Leerzeichen ersetzt werden.

2.Beispiel: Das Setzen einer Bearbeitungsregion (ROI) wird folgendermaßen durchgeführt:

```
roi_size ( 512, 512, 0, 0 )
```

Hiermit wird die aktuelle ROI auf die Werte x1 = 0, y1 = 0 (das ist die linke obere Ecke des Bildschirmes) bei einer Breite und Höhe von jeweils 512 Pixeln gesetzt. Vereinfacht, mit Leerzeichen als Delimiter, kann der Befehl auch folgendermaßen geschrieben werden: roi_size 512 512 0 0. Selbstverständlich können statt der Konstanten auch Variable stehen:

```
DEFINT Breite
DEFINT Hoehe
DEFINT X1
DEFINT Y1
Breite := 512
Hoehe := 512
X1 := 0
Y1 := 0
roi_size ( Breite, Hoehe, X1, Y1 )
```

Die Reihenfolge und Anzahl der Parameter muß genau eingehalten werden. Sie ist in der Beschreibung aller Funktionen angegeben. Der Macro-Interpreter überprüft die Anzahl der Parameter beim Funktionsaufruf. Wenn zuwenig Parameter angegeben wurden, so wird eine Warnung ausgegeben und dann die fehlenden Parameter zu Null angenommen. (Wenn "msg_interactive" auf Null gesetzt wurde, wird diese Warnung natürlich unterdrückt) Bei Single_Step oder Test_Macro wird der Befehl "msg_interactive" nicht ausgeführt, so daß alle Meldungen sichtbar werden und quittiert werden müssen. Es folgt eine Liste der Themenbereiche, zu denen Funktionen vorliegen.

Allgemeine Kontrollfunktionen:	Systemfunktionen und Macro-Kontrolle Buffer Initialisierung, Einstellung und Transfer ROI (Region of Interest) Einstellung Input und Output Funktionen DOS Funktionen Lade- und Speicherfunktionen Farbtabellensteuerung Printeransteuerung Kopier- und Verschiebefunktionen Löschroutinen Bildübernahmefunktionen (Frame Grabbing)
Bildvorverarbeitung (Processing):	Geometrische Korrekturen Histogrammausgleich Arithmetische Funktionen Lineare und nichtlineare Filter Echtfarbbearbeitung Frequenzraumbearbeitung Texturerkennung Pyramidenerzeugung Binärisierung Morphologie 3D/Sequenz-Bildverarbeitung (Zusatzmodul)
Bildanalyse (Measurement):	Geometrische Kontrollfunktionen Geometrische Kalibrierung Geometrische Vermessungen Statistische Vermessung Edge Detection Object Classification Stereometrische Vermessung (Zusatzmodul)

	Pattern Recognition
	Particle Image Velocimetry (Zusatzmodul)
	Interferometrie (Zusatzmodul)
	Fringe Correlation (PGM) (Zusatzmodul)
	Echtzeit-Positionsbestimmung (Zusatzmodul)
	Optical Character Recognition (OCR) (Zusatzmodul)
	BarCode Recognition (Zusatzmodul)
Graphik:	Testbilderzeugung
	Graphikfunktionen
	Texteinbindung

Alle Funktionen liefern mindestens einen Wert zurück - den Return-Value. Dieser Wert ist bei normaler Funktionsausführung immer +1. Im Fehlerfall, d. h. wenn die Funktion infolge einer Fehlerbedingung abbricht, wird -1 zurückgeliefert. Dies ist z. B. der Fall, wenn die Funktion aufgrund von Speichermangel oder bei Eingabe unzulässiger Parameter nicht ausgeführt werden kann. Die -1 bewirkt normalerweise einen Abbruch der Macrofunktion, da eine Fortsetzung üblicherweise nicht sinnvoll ist. Dieser Abbruch kann jedoch verhindert werden, indem die Funktion "stop_on_error (0)" aufgerufen wird. Der Return-Value wird in die Variable IVAR[0] gespeichert und kann dann z. B. als Entscheidungsgrundlage für einen anderen Macrofunktionsablauf genutzt werden. In bestimmten Fällen liefern einige Funktionen den Wert 0 zurück, wenn es sich z. B. um einen zugelassenen Funktionsabbruch - z. B. in einer Dialogbox - handelt. In diesem Fall wird normalerweise eine Warnung ausgegeben, die jedoch auch abgeschaltet werden kann.

Viele Funktionen - z. B. Meßfunktionen - liefern Meßergebnisse zurück, die im Macroprogramm zur weiteren Bearbeitung genutzt werden können. Diese Werte werden in den Variablen IVAR[10] bis IVAR[19] und FVAR[10] bis FVAR[19] sowie in der Stringvariablen STRING[1] zurückgegeben. Wenn sie über längere Zeiträume genutzt werden sollen, so empfiehlt es sich, sie durch Kopieren auf andere Variable vor einem Überschreiben durch nachfolgende picCOLOR-Funktionen zu schützen.

Im "User"-Menü finden Sie die folgenden (oder zumindest eine Auswahl davon) Funktionen, mit denen Sie Macro-Programme starten und austesten können:

Fehlerbehandlung

Wie oben beschrieben, können MACRO-Funktionen einen Fehler zurückliefern und können dann zum Abbruch des MACRO-Programmes führen. Noch gravierendere Fehler sind allerdings Funktionen oder Kommandos, die überhaupt nicht ausgeführt werden können, weil sie entweder in der benutzten picCOLOR-Programmversion nicht existieren oder falsch geschrieben sind. Dies gilt auch für falsch buchstabierte Variablennamen, die also nicht der Definition entsprechen. In diesem Fall bricht das MACRO sofort ab und gibt den fehlerhaften Befehl im Klartext aus, incl. der kompletten Parameterzeile, so daß eine Fehlersuche einfach vonstatten gehen sollte. Andere Fehler, z. B. nicht richtig geschachtelte FOR-, WHILE-, oder IF-ELSE-Konstruktionen werden in der Regel erkannt und angemahnt. Dies gilt auch für im Kontext nicht erlaubte Operatoren.

Record Macro

MACRO-Funktionen werden mit dem eingebauten Editor oder mit einem beliebigen Text-Editor geschrieben. Dies bringt den Vorteil mit sich, daß auch bestimmte Kontrollstrukturen (z. B. FOR-Loops und WHILE-Schleifen oder Entscheidungen mit IF-ELSE-ENDIF) programmiert werden können. Es muß beachtet werden, daß der MACRO-Funktionsfile die Extension "MCR" besitzt. In zukünftigen Versionen ist geplant, mit der "Record"-Funktion ein MACRO auch automatisch während einer interaktiven Bildverarbeitungssitzung aufnehmen zu können. MACRO-Programme können praktisch beliebig lang sein (bis zur Grenze des internen Heap-Speichers). Z.Zt. wird jedoch beim Laden eines MACRO-Programmes nur 64 kBytes an Speicherplatz freigegeben. Dies kann auf Wunsch beliebig geändert werden.

Load & Start Macro

Hier kann ein MACRO-Programm in einer Dialogbox ausgewählt, von Disk geladen und gestartet werden. MACRO-Programme sind ASCII-Textfiles mit der Extension .MCR. Die gleiche Funktion ist in der Tool-Box verfügbar.

Show Only

Mit dem "Show Only"-Schalter kann der Ablauf eines MACRO-Programmes getestet werden Es werden die Namen und Parameter aller Kommandos und Funktionen auf dem Bildschirm ausgegeben. Sie werden jedoch nicht ausgeführt. In bestimmten Fällen, wenn z. B. die Programmausführung von einer eingefragten Input-Variable oder vom Ergebnis einer Funktion abhängt, kann der Test natürlich nicht den wahren Macro-Ablauf wiedergeben.

Single Step

Bei Einschalten des "Single Step"-Schalters wird vor der Ausführung jedes Kommandos oder jeder Funktion zunächst wie bei der Funktion "Show Only" der Name und die Parameter angezeigt. Nach Bestätigung wird die Funktion dann ausgeführt.

Message-Time

Viele Funktionen geben kurze Meldungen zur Information des Benutzers über eine erfolgreiche oder nicht erfolgreiche Ausführung aus. Diese müssen normalerweise vom Benutzer mit Maus oder Tastatur bestätigt werden. Dies würde natürlich einen vollautomatischen Ablauf eines MACRO-Programmes stören. Daher kann eingestellt werden, daß die Meldungen automatisch nach bestimmter Zeit verschwinden. In diesem Fall werden sie immer automatisch positiv quittiert.

Copy to Note

Wenn der "Copy_to_Note"-Schalter eingeschaltet ist, wird das eingelesene und vorkompilierte MACRO-Programm zur späteren Ansicht und Inspektion in den Notebook- (gleich dem MACRO-Editor)-Speicher kopiert. Hierbei ist zu beachten, daß ein Macro-Programm während des Ladens "vorkompiliert" wird, um einen wesentlich schnelleren Programmablauf zu gewährleisten. Hierbei werden die Funktionsnamen und eventuell auch die Kommando-Namen speziell kodiert. Alle Funktionsnamen erhalten eine Kurzbezeichnung mit dem Buchstaben "F" und daran anschließend einer laufenden Nummer, nämlich der Nummer in der Macro-Funktionstabelle. Vorsicht: da der Notebookspeicher z. Zt. in der Länge begrenzt ist, kann es sein, daß das Macro-Programm abgeschnitten wird und dann nicht mehr lauffähig ist. Diese Funktion ist nur in Testversionen implementiert und eigentlich nur für die Fehlersuche bei der Entwicklung des Macro-Interpreters gedacht.

Edit Macro

Mit einem der unter Windows zur Verfügung stehenden Editoren kann ein Macro-Programm erstellt und editiert werden. In bestimmten Versionen des picCOLOR-Programmes ist ein eigener Macro-Texteditor eingebaut, der mit dieser Funktion aufgerufen werden kann. Hier ist die automatisch benutzte Extension für Macro-Programmfiles "MCR". Sie braucht beim Laden oder Abspeichern nicht angegeben zu werden.

Macro-Funktionstabelle (Function Table)

Hier wird eine Tabelle gezeigt, in der alle verfügbaren Macro-Funktionen (z. Zt. jedoch nicht die Funktionen des USER-Modules), geordnet nach Themenbereichen aufgeführt sind. Die Art und Anzahl ihrer Parameter wird durch die Abkürzungen "i1..i4, f1..f2, string1" angedeutet. Zusätzlich werden die Namen der Variablen explizit angegeben. Sie können eine Hilfe bei der Bedienung der Macro-Funktionen sein und ersetzen in einfachen Fällen das Macro-Handbuch.

Macro Running (Flag)

Dieses Flag zeigt an, ob gerade ein Macro-Programm augeführt wird. Da während der Macro-Ausführung in der Regel die interaktive Eingabe über die Maus und die Tastatur möglich ist, kann also während eines Macro-Ablaufes in bestimmter Weise eingegriffen werden. In der Regel ist dies nicht vorgesehen und nicht gestattet, außer in Notfällen. So kann z. B. durch Anklicken des "Macro-Running"-Flags die Macro-Ausführung angehalten werden. Hierbei muß beachtet werden, daß ein Macro bereits angehalten wird, wenn interaktiv eines der picCOLOR-Menüs aufgerufen wird. Erst wenn dieses geschlossen wird, läuft das Macro-Programm weiter. Diese interaktive Eingriffsmöglichkeit kann mit der Funktion "user_interaction" geschaltet werden.

Send Macro Function

Hier öffnet sich eine Texteingabe-Box, in der ein Macro-Kommando mit Parametern eingegeben werden kann und ausgeführt werden kann. Mit dieser Funktion können spezielle Macro-Funktionen gestartet werden, die z. B. nicht über das Menü erreicht werden können oder bestimmte Parameter gesetzt werden, die nicht über das Menü gesetzt werden können.

Show Variables

Es werden die Macro-Übergabevariablen IVAR[10] bis IVAR[19] und FVAR[10] bis FVAR[19], außerdem der STRING[1] und der Return Value in IVAR[0] im aktuellen Zustand angezeigt. Außerdem wird die zuletzt ausgeführte Funktion in STRING[0] angezeigt.

Funktionsbeispiel: Top-Hat-Funktion

TOPHAT.MCR (simplified version):

```
DEFINT  ActiveImage                 /* define integer variables */
DEFINT  SecondImage
DEFINT  Openings
DEFINT  DefinedNewBuffer
transfer_to_undo -1                 /* active Image to Undo Buffer */
enable_undo 0                       /* disable automatic undo function */
get_active
ActiveImage := IVAR[10]             /* active image buffer to "activeimage" */
stop_on_error  0                    /* no abort at error */
msg_interactive 0                   /* do not show any messages */
define_imagebuf 0                   /* define temporary image buffer... */
IF (IVAR[0]  =  -1)                 /* in case of memory error use available buffers */
    stop_on_error 1                 /* reset error handling */
    msg_interactive -1              /* reset messages handling */
    IF (ActiveImage  #  2)
        user_message ''Image buffer 2 will be used!''
        SecondImage := 2        /* ...use buffer 2 or... */
    ELSE
        user_message ''Image buffer 1 will be destroyed!''
        SecondImage := 1        /* ...use buffer 1 */
    ENDIF
    DefinedNewBuffer := 0           /* flag = 0, no new buffer has been defined */
ELSE                                /* intermediate buffer definition ok */
    stop_on_error 1
    msg_interactive -1
    get_last_buffer                 /* get just defined buffer number in IVAR[10] */
     SecondImage := IVAR[10]
     DefinedNewBuffer := 1          /* and set flag to 1, new buffer has been defined */
ENDIF
long_input ''How many opening iterations [1..5]?''   /* Return Value in IVAR[10] */
Openings := IVAR[10]
IF Openings  >  0
    transfer_buffer (ActiveImage, SecondImage)    /* copy to temporary buffer */
    select_active (0, SecondImage)                /* activate temporary buffer */
    morph_e_d_o_c (2, 1, 5, Openings)             /* circular opening n times */
    select_active (0, ActiveImage)
    select_second (0, SecondImage)
    src_dst_op (2, 0)                             /* and subtract from original */
ELSE
    error_box ''Iterations must be > 0''
ENDIF
IF DefinedNewBuffer = 1             /* undef buffer, if one has been defined before */
    msg_interactive 0
    free_imagebuf  0                /* deallocate temporary buffer, do not show any messages here */
    msg_interactive -1
ELSE
    SKIP                            /* do nothing if already defined buffer 1 or 2 were used */
ENDIF
enable_undo 1                       /* and enable undo function */
```

Zuerst wird versucht, in einem neu definierten temporären Bildspeicher zu arbeiten. Nur, wenn dies zu einer Fehlerbedingung führt - sei es, daß nicht genügend Speicher zur Verfügung steht oder daß schon die maximal zugelassene Anzahl von Bildspeichern definiert ist - wird ein vorhandener Buffer benutzt, nämlich Buffer 1 oder 2. Nach Eingabe der Iterationsanzahl wird die TOPHAT-Funktion ausgeführt und schließlich der temporäre Buffer, falls er definiert wurde, wieder freigegeben.

Funktionsbeispiel: Bildübernahme

Eine normale Bildeinlesesequenz aus einem MACRO-Programm wird z. B. folgendermaßen programmiert (bei der älteren picCOLOR V2.17 Karte muß nach setup_capture wegen der neuen Synchronisation auf 50 Hz eine Sekunde gewartet werden; aber auch bei modernen Frame Grabbern kann eine kurze Wartezeit sinnvoll sein):

```
setup_capture          /* initialize all acquisition modules and acquisition memory */
time_delay 1000        /* 1 Sek. Einschwingzeit für die Hardware... */
cam_sync_inp 1 1       /* set camera and sync inputs (F64 and picCOLOR V2.17 only) */
...
...                    /* other program */
...
acquire 0 1 0          /* grab single image (snap shot)  */
...
...                    /* other program */
...
close_capture          /* end frame grabbing */
time_delay 1000        /* 1 Sek. Einschwingzeit für die Hardware... */
```

Statt der einfachen "snap_shot"-Funktion könnte man auch die folgende Sequenz

```
DEFINT    Right_Mouse_Button
acquire 0 -1 0                    /* continuous grabbing on */
...
...
...
Right_Mouse_Button := 0
WHILE   Right_Mouse_Button # 1
   get_mouse_button               /* right mouse button will set IVAR[11] to "1" */
   Right_Mouse_Button := IVAR[11]
ENDWHILE
acquire 0 0 0                     /* continuous grabbing off */
...
...
```

verwenden. Zwischen "acquire 0 -1 0" und "acquire 0 0 0" kann hiermit interaktiv z. B. auf einen Event gewartet werden. Über Funktionen, die eine Schnittstelle auslesen, kann dies natürlich auch automatisch getriggert geschehen.

Macro-Interpreterstruktur und Syntax

Syntax der MACRO-Sprache in der Backus-Naur Form (BNF):

```
block =          command { command }.
command =        [ DEFINT  stringconstant crlf |
                 DEFFLT  stringconstant crlf |
                 IVAR''[''expression'']'' integerassignmentoperator expression crlf |
                 FVAR''[''expression'']'' floatassignmentoperator expression crlf |
                 STRING''[''expression'']'' stringassignmentoperator stringexpression crlf |
                 FOR ''(''expression'')'' crlf  block ENDFOR crlf |
                 WHILE ''(''condition'')'' crlf block ENDWHILE crlf |
                 IF ''(''condition'')'' crlf  block ELSE crlf block ENDIF crlf |
                 SKIP crlf |
                 STOP crlf |
                 CALL ''expression'' crlf |
                 FUNCTION ''expression'' crlf  |
                 RETURN crlf |
                 comment crlf |
                 piccolorfunction ''(''parameter '')'' crlf ].
condition =      expression [''='' | ''#'' | ''<'' | ''>'' | ''<='' | ''>=''] expression.
expression =     [ integer | floatingpoint | IVAR''[''expression'']'' | FVAR''[''expression'']'' ].
stringexpression = [ ''''''stringconstant''''''' |
                 STRING''[''expression'']'' |
                 CONCAT ''('' stringexpression'','' stringexpression'','' expression '')'' |
                 LTOA ''('' expression '')'' |
                 ITOA ''('' expression '')''  |
                 FTOA ''('' expression '')'' |
                 ETOA ''('' expression '')'' |
                 STRCMP ''(''stringexpression'',''stringexpression'')'' ].
integerassignmentoperator =    [''':='' | ''+='' | ''-='' | ''*='' | ''/='' | ''<<='' | ''>>=''
                                                                | ''&='' | ''|='' | ''^='' | ''%=''].
floatassignmentoperator =      [''':='' | ''+='' | ''-='' | ''*='' | ''/='' ].
stringassignoperator =     ''':=''.
comment =          [ ''/*'' stringconstant ''*/'' | ''*'' (in erster Spalte) stringconstant ].
crlf =             [ Carriage Return | Linefeed ].
piccolorfunction = [ picCOLOR Funktion nach weiter unten folgender Auflistung ]
parameter =        [ expression { expression } | stringexpression ].
stringconstant =   [ character{character} ].
integer =        [sign]number { number }.
floatingpoint = [sign]number {number} ''.'' number {number} [''E''[sign]number{number}].
sign =           [''+'' | ''-'' ].
number =         [''1'' | ''2'' | ''3'' | ''4'' | ''5'' | ''6'' | ''7'' | ''8'' | ''9'' | ''0''].
```

Macro-Funktionslistenstruktur

Es folgt nun eine kurze Auflistung aller zur Verfügung stehenden Macro-Funktionen, aufgeteilt in Themenbereiche. Anhand der zusätzlich angegebenen Kapitelnummern kann die genauere Beschreibung der einzelnen Funktionen im picCOLOR-Handbuch nachgelesen werden. Hinter dem Funktionsnamen steht die Anzahl und der Typ der Parameter, die zu der jeweiligen Funktion übergeben werden müssen, als Kennzahl nach der folgenden Tabelle kodiert:

0 : kein Parameter
1 : 1 Integer
2 : 2 Integer
3 : 3 Integer
4 : 4 Integer
5 : 1 Floating Point
6 : 2 Floating Point
7 : 1 Integer, 1 Floating Point
8 : 2 Integer, 2 Floating Point
9 : Characterstring
10 : komplexere Parameterstruktur (z. Zt. noch nicht implementiert)

Zusätzlich ist dort noch ein Kennbuchstabe angegeben, der aussagt, ob die Funktion interaktiv oder nicht interaktiv (Batch) ausgeführt wird. Interaktive Funktionen, bei denen z. B. eine Eingabe vom Benutzer verlangt wird, sind nicht geeignet, um vollautomatisch ablaufende Macros zu erstellen. Es gibt drei verschiedene Arten von interaktiven Funktionen: bei der Kennung ‘‘ID’’ erhält man eine graphische Dialogbox, in der vor Ausführung der Funktion z. B. mit einigen Schaltern Parameter zu dieser Funktion gesetzt werden können. Die Kennung ‘‘IM’’ bedeutet, daß eine Meldung auf dem Bildschirm ausgegeben wird, die vom Benutzer mit einer Maustaste quittiert werden muß. Die dritte Gruppe von interaktiven Funktionen läßt sich in keine der vorgenannten Gruppen einordnen und erhält einfach die Kennung ‘‘I’’ für interaktiv. Diese Funktionen erwarten interaktive Eingaben unterschiedlicher Art, z. B. eine Pixelanwahl mit einem Fadenkreuz. Vollautomatisch ablaufende Funktionen erhalten die Kennung ‘‘B’’ für ‘‘Batch-Funktionen’’. Funktionen, die durch die Kennung ‘‘B (IM)’’ gekennzeichnet sind, sind eigentlich Batch-Funktionen, können jedoch unter bestimmten Bedingungen Messages erzeugen, z. B. im Fehlerfall eine Fehlermeldung, die dann wiederum vom Benutzer quittiert werden muß. Nach Auftreten von Fehlern wird die Ausführung der Macro-Funktion üblicherweise abgebrochen.

Funktionen mit der Kennung ‘‘IM’’, also Funktionen, die nur eine Meldung auf den Bildschirm bringen, können mit der Funktion ‘‘msg_interactive (0)’’ zu Batch-Funktionen geändert werden. Die Meldungen werden dann automatisch unterdrückt. Durch Aufrufen der Funktion ‘‘msg_interactive (-1)’’ werden diese Funktionen wieder interaktiv und müssen vom Benutzer quittiert werden. Dies gilt auch für Batch-Funktionen, die z. B. im Fehlerfall eine Meldung ausgeben (Kennung: B(IM)). Ein Parameter n>0 läßt eine interaktive Meldung n Sekunden auf dem Bildschirm. Wenn die Meldung automatisch weggeschaltet wurde, wird sie immer positiv behandelt, d. h. wie mit dem OK-Schalter (oder in bestimmten Fällen auch mit der linken Maustaste) bestätigt.

Schließlich werden in der Liste alle Parameter, die der Funktion übergeben werden müssen, mit Namen aufgeführt und es wird eine kurze Funktionsbeschreibung gegeben. Funktionsnamen werden immer klein geschrieben. /* In Kommentarklammern gesetzte Funktionen sind nicht oder noch nicht implementiert.*/ Es werden auch alle Funktionen der picCOLOR-Zusatzmodule zumindest sehr kurz aufgeführt, auch wenn diese je nach Ausstattung des Programmes nicht ausführbar sind.

Auf den Programmdisketten oder der Programm-CD finden Sie noch einige Beispiele zu einfachen Macro-Funktionen, die die Anwendung der Kommando- und Funktionssprache verdeutlichen.

Inhaltsverzeichnis Macro-Funktionen

Systemfunktionen	19
Macrokontrollfunktionen	23
Buffer Initialisierung	25
Bildschirm und Bildspeicher	26
ROI (Region of Interest) Einstellung	28
Input und Output Funktionen	29
DOS Funktionen	33
Lade- und Speicherfunktionen	34
Farbtabellensteuerung	37
Printeransteuerung	39
Kopier- und Verschiebefunktionen	40
Löschroutinen	41
Bildübernahmefunktionen	42
Geometrische Korrekturen	48
Histogrammausgleich	51
Arithmetische Funktionen	52
Lineare und nichtlineare Filter	53
Echtfarbbearbeitung (z. T. Zusatzmodul)	57
Frequenzraumbearbeitung	59
Pyramidenerzeugung	62
Texturerkennung	63
Binärisierung	64
Morphologie	67
3D/Sequenz-Bildverarbeitung (Zusatzmodul)	69
Geometrische Kontrollfunktionen	73
Geometrische Kalibrierung	74
Geometrische Vermessungen	76
Statistische Vermessung	85
Edge Detection	88
Object Classification	89
Stereometrische Vermessung (Zusatzmodul)	90
Pattern Recognition	91
Particle Image Velocimetry (Zusatzmodul)	93
Interferometrie (Zusatzmodul)	100
Fringe Correlation (PGM) (Zusatzmodul)	102
Positionsbestimmung 2D/3D (Zusatzmodul)	111
Echtzeit-Funktionen (Real Time) (Zusatzmodul)	117
3D-Rekontruktion (Zusatzmodul)	119
Optical Character Recognition (OCR) (Zusatzmodul)	123
BarCode Recognition (Zusatzmodul)	124
Neural Network (Zusatzmodul)	125
Graph Theory (Zusatzmodul)	126
Testbilderzeugung	127
Graphik- und Texteinbindung	128
Anhang: Fehlermeldungen	130

Funktionen und Parameter

/* In Kommentarklammern gesetzte Funktionen sind nicht oder nicht vollständig implementiert
und können noch nicht aufgerufen werden */

Systemfunktionen (Manual Kapitel 6.12. sowie Toolbox in Kapitel 13.):

reset_system	1, B/ID	interactive: integer; Zurücksetzen des picCOLOR-Systemes auf einen mehr oder weniger definierten Anfangszustand. Alle Bildspeicher werden gelöscht. Der FrameGrabber wird ebenfalls zurückgesetzt. Dies ist jedoch stark hardware-abhängig. Interne Einstellungen und Variable bleiben jedoch zum größten Teil bestehen. Wenn "interactive" zu 1 gesetzt ist, erscheint die picCOLOR-Start-Dialogbox, in der dann eine Bildspeichergröße gewählt wird und ein oder mehrere normale Bildspeicher definiert werden können.
version_nr	1, B/ID	interactive: integer; Angabe der Versionsnummer des picCOLOR-Programmes interactive = 0: Ausgabe im Macro-String[1] interactive = 1: Ausgabe in einer Dialogbox
load_status	0, ID,	kein Parameter; Laden einer Systemkonfiguration (.INI-File). Return_Value ist 1 bei einwandfreier Ausführung der Funktion, 0 bei User-Abbruch, -1 bei File-Error.
save_status	0, ID,	kein Parameter; Speichern einer Systemkonfiguration (.INI-File). ReturnValue ist 1 bei einwandfreier Ausführung, -1 bei File-Error.
conf_params	0, ID,	kein Parameter; Anzeige bestimmter Systemparameter und Laden und Speichern der .INI-Files. ReturnValue immer 1.
system_time	1, B/IM	interactive: integer; Anzeige oder Abfrage der Systemzeit von picCOLOR. Bei interactive=1 wird in einer Message-Box die Systemzeit seit Programmstart in Sekunden und außerdem die Host-Zeit (die Real Time Clock des PC's) angezeigt. Wenn interactive=0, wird die picCOLOR-Systemzeit in der Variable FVAR[10] zurückgegeben. Bei interactive=1 wird die Host-Zeit in den Integervariablen [10..17] - sec, min, hour, day, month, year, dayofweek, millisec - und komplett im String[1] zurückgegeben (s.u.). Return_Value immer 1.
host_time	4, B,	wait, hour, min, sec: integer; Abfrage der Real-Time-Clock des Host-Rechners und eventuelles Warten auf einen bestimmten Zeitpunkt. wait = 0: get host time wait = 1: wait for host time to match parameters hour, min, sec (can be interrupted by ESC or right mouse button) wait = 2: same as (wait=1), but no interrupt possible. Bei wait==1 oder 2 kann der Zeitpunkt wie folgt eingegeben werden: hour: [0..23] min: [0..59] sec: [0..59] Übergabe der Ergebnisse wie folgt: IVAR[10] = sec [0..59] IVAR[11] = min [0..59] IVAR[12] = hour [0..23] IVAR[13] = month day [1..31] IVAR[14] = month [0..11] IVAR[15] = year since 1900 IVAR[16] = week day [0..6] IVAR[17] = Millisekunden STRING[1] = Date/Time [m/d/y, h:m:s]: .....
highres_timer	1, B	query_setup: integer; Abfrage des High-Resolution Timers des Systems. Er ist z. Zt. auf 1/10 Millisekunden Auflösung eingestellt. query_setup = 0|1: 1=setup timer, 0=query timer IVAR[10] = time STRING[1] = Time-String
time_delay	1, B	time: integer: Wartefunktion, picCOLOR legt sich schlafen. time in Millisekunden > 20, auf etwa 16 ms genau, da der interne Windows-Timer mit dieser Frequenz läuft. IVAR[10] = 0, außer wenn mit ESC abgebrochen wurde, dann IVAR[10] = 1
help	0, ID,	kein Parameter; Anzeige einer Hilfe-Meldung. Return_Value immer 1. (nicht impl.)
hot_key	1, B,	key: integer-Äquivalent des Hot-Key-Characters; Ausführung einer Funktion, die normalerweise mit einem Hot-Key erreicht wird. Beispiel: hot_key (82) zeigt den ROT-Anteil, da 82 = hex(52) = ''R''. hot_key(85) ist "Undo". ReturnValue immer 1.
trig_funct	7, B	type: integer, val: float; Berechnung einer trigonometrischen Funktion und Rückgabe des Ergebnisses in FVAR[10]. Return-Value immer 1.

		type: 0 val 5 acos(val) 1 sin(val) 6 atan(val) 2 cos(val) 7 sinh(val) 3 tan(val) 8 cosh(val) 4 asin(val) 9 tanh(val) Alle Werte werden gegebenenfalls in Radianten erwartet und zurückgegeben.
log_funct	7, B	type: integer, val: float; Berechnung einer logarithmischen Funktion und Rückgabe des Ergebnisses in FVAR[10]. Return-Value immer 1. type: 0 val 1 exp(val) 2 log(val) 3 log10(val) 4 sqrt(val) 5 fabs(val) 6 (double)round(val)
str_funct	4, B	type, string, index, val: integer; Funktionen zur Bearbeitung von STRING[]-Variablen: type = 0: IVAR[10] = strlen(STRING[string]) type = 1: IVAR[10] = (int)STRING[string][index] type = 2: STRING[string][index] = val (if index < strlen(string)) type = 3: Append (val) characters from String[index] to String[string] Return Value ist 1, außer wenn bei type=2 der Index zu groß ist, dann -1
notebook	1, ID/B,	type: integer; Notebook-Funktion: type = 0: In einer interaktiven Dialogbox können hier kurze Texte interaktiv eingegeben werden. Diese können interaktiv auch geladen oder abgespeichert werden oder zusammen mit Bilddaten im TIFF-Bildfile abgespeichert werden. type = -1: Der Notebook-Speicher wird gelöscht. type = 1: Der Inhalt des STRING[1] wird an den Notebook-Speicher angehängt. type = 2: load from open file type = -2: load interactive type = 3: save to open file type = -3: save interactive Return_Value immer 1, außer bei File-Error. Achtung: es gibt eine maximale Länge des Notebooktextes, darüber wird abgeschnitten.
edit_data_file	0, ID,	kein Parameter: Editieren eines Datenfiles in einer Text-Dialogbox. (noch nicht implementiert, nur für picCOLOR V2.17)
dio_counter	4, B/I	type, prm1, prm2, prm3: integer; Setzen eines externen Timers/Counters: type = 0: open counter/dio-Karte (prm1 = device number) type = 1: config counter (always interactive) type = 2: close counter/dio-Karte type = 3: start counter: prm1=id, prm2=mode, prm3=rate type = 4: stop counter (not implemented) type = 5: read active counter to IVAR[10] type = 6: Set Hardware Trigger I/O Line Number (z. B. 0..47 auf Digital-I/O-Karte, 100..199 Acq-Karte) type = 7: Set Digital I/O Line: prm1 = I/O-Line, prm2 = Set Bit: 0|1 type = 8: Get Digital I/O Line: prm1 = I/O-Line, IVAR[10] = Get Bit: 0|1 type = 9: Set Counter ID: prm1 = Counter-ID = 0|1|2 bei der DII-Karte, oder 0|1|2|3 bei den National-Instruments-Karten mit dem NI-DAQmx-Treiber Rate und Mode (oder Freq und Dutycycle bei den NI-Karten) werden dann aus den durch type=3 vorbelegten Variablen gesetzt. type = 10: DIO-first-IN = prm1, DIO-last-IN = prm2 (für NI-Karten) type = 11: DIO-first-OUT = prm1, DIO-last-OUT = prm2 (für NI-Karten) type = 12: Definition des DIO-Device-Namens aus STRING[prm1]. Für die DII-8255-Karte gilt für die alte Treiber-API der Name "*Device", für die neue Treiber-API der Name "PCI_8255". Der neue Name muss unter Vista oder Win7 oder höher benutzt werden, der alte Name wird für Win2000 oder WinXP benutzt. type = 100: TicksSourceExt100kHz für NI-Karten. prm1=0=ext. source, prm1=1=100kHz internal source Achtung: Bei der DII I/O-Karte wird mit type=7 bei der ersten Benutzung automatisch der komplette Port (8 Bit) auf Output geschaltet, bei type = 8 auf Input. Dies kann dann

		erst durch close und erneutes open wieder geändert werden. (Siehe auch die set_trigger-Funktion und DII-Handbuch) Bei den National-Instruments-Karten, z.B. der NI-6320X, werden Input und Output-Range über die dio_counter-Funktion mit den Parametern type=10 und type=11 gesetzt.
smc_control	4, B/I	type, xStep, yStep, zStep: integer; Schrittmotorsteuerung, sehr hardwareabhängig... Hier implementiert für die Schrittmotorsteuerkarten SMC800 oder SMC1500 der Firma EMIS - Ges. für Elektronik und Mikroprozessorsysteme mbH, Zur Drehscheibe 4, 92637 Weiden, vertrieben durch Conrad Elektronik GmbH. Auf Anfrage sind Macro-Programme zur Mikroskop-Steuerung verfügbar! Achtung: Port muß geöffnet sein! type = 0: close type = 1:init, x,y,z: Referenzfahrt Schritte type = 2: x = StartFrequenz, y=Arbeitsfrequenz type = 3: x = Beschleunigung, y=Bremsen type = 4: x,y,z = Vektor type = 5: Test, interactive Cursor Shift*10
dde_transfer	2, B	picc_to_dde_off_on, dde_read_from_io: integer; Kontrolle der DDE-Schnittstelle für einen Transfer vom und zum picCOLOR-Programm picc_to_dde_off_on = (-1|0|1): Ein- und Ausschalten eines Transfers von picCOLOR zur DDE-Schnittstelle. (-1: don't change) dde_read_from_io = (-1|0|1): Wenn eingeschaltet, liest picCOLOR DDE-Befehle von der I/O-Box. Diese Befehle müssen den Standards des picCOLOR-MACRO-Interpreters folgen. Es sind allerdings z. Zt. nur Funktionsbefehle zugelassen, keine Kontrollstrukturen oder Variablenbesetzungen. Die Kommandos muessen über einen POKE auf das Item "IOString" zum picCOLOR-Programm gesandt werden. Rückgabewerte können über das Item "MResult" abgefragt werden. Dabei werden die Daten im folgenden Format übertragen: ReturnValue: 1 IVAR[10..19]: 0 0 0 0 0 0 0 0 0 0 FVAR[10..19]: 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 STRING[1]: stringvariable Hierzu muß der Transfer mit picc_to_dde_off_on==1 eingeschaltet sein. Die Eingabe von (-1) für einen oder beide Parameter bewirkt keine Änderung. ReturnValue ist immer 1.
tcpip_server	2, B	close_open, socket: integer; Open or Close a TCP/IP Socket. A non-blocking TCP/IP connection (socket) will be established. Data and commands can be transfered via this connection. close_open: -1: close listening socket, if all other connections are closed close_open: 0: close socket close_open: 1: open socket socket: for close operations: -1: close both (all) sockets; 0|1: close socket 0|1 socket: for open operations: 0|1 for opening socket 0|1
tcpip_params	4, B/ID	type, p1, p2, p3: integer; Set TCP/IP parameters: (parameter==-1: no change!) type = -1: open Dialogbox to set all parameters type = 0: Get last error in IVAR[10] type = 1: Address := STRING[p1], for example: "127.0.0.1" for system Port := STRING[p2], for instance 27015 Returns: IVAR[10] = port, STRING[1] = address type = 2: p1 = tcpip-timer: 0=off, [1..n]=send timer strings in milliseconds p2 = 0|1: transfer 2d-data p3 = 0|1: transfer 3d-data If both 2d and 3d transfers are switched on, 3d data comes first, then 2d data, because 3d-algorithms may reconstruct missing 2d points. Returns: IVAR[10] = timer, IVAR[11] = trans2d, IVAR[12] = trans3d type = 3: p1 = 0|1: read commands from I/O box, feature off/on: alle picCOLOR-Funktionen und die IVAR-, FVAR-, STRING-Zuweisung sowie STRCMP können gesendet werden. Wenn keine Error-Meldungen erwünscht sind, mit msg_interactive(0) abschalten. p2 = 0|1: transfer_io: enable to send all other I/O via TCP/IP p3 = 0|1: transfer_piv: enable to send all PIV data via TCP/IP Returns: IVAR[10] = read_io, IVAR[11] = trans_io, IVAR[12] = trans_piv type = 4: p1 = 0|1: use TCP = 0 or UDP = 1 Interface Returns: IVAR[10] =tcp_udp

arrange_window	3, B	type, x_width, y_height: integer; Größen- und Lageänderung der Bildfenster innerhalb des picCOLOR-Hauptfensters. Außerdem kann auch das picCOLOR-Hauptfenster verändert werden. type = 0: Actual Child Window: Position x,y type = 1: Actual Child Window: Size width,height type = 2: Actual Child Window: Maximize and Position x,y type = 3: Main Window: Position x,y type = 4: Main Window: Size width,height type = 5: Main Window: Maximize type = 6: Maximize all Windows type = 7: Pan/Scroll active window by 0 <= x,y <= max
zoomout_window	2, B/I	inter_toggle, zoomout: integer; Zur verkleinerten Darstellung großer Bilder (Window) können diese auf den ZoomOut-Modus geschaltet werden. Wenn dann das Bildfenster verkleinert wird, wird immer der gesamte Bildinhalt dargestellt, gegebenenfalls stark verkleinert. Defaultmäßig wird das Seitenverhältnis beibehalten, es kann jedoch auch frei gewählt werden. inter_toggle = 1: toggle, ask interactively when switching on inter_toggle = 0: set zoomout mode for actual window to 0|1 inter_toggle = -1: get zoomout mode for actual window to IVAR[10] = 0|1 zoomout = 0|1|2 = off | keep aspect ratio | do not keep aspect ratio IVAR[10] = Zoomout-Mode = 0|1, IVAR[11] = KeepAspectRatio = 0|1
load_pmask	0, ID,	kein Parameter; Eingabe des Pixel-Mask-Registers für Bildschirmdarstellungen. Return_Value immer 1. (nicht implementiert, nur für picCOLOR V2.17 Hardware)
scr_wait_blank	0, B	kein Parameter: Warten auf das Vertical Blanking des Bildschirmes, um z. B. bestimmte Events auf den Bilddurchlauf zu synchronisieren. (noch nicht implementiert, nur für picCOLOR V2.17 Hardware)

Macrokontrollfunktionen (Manual Kapitel 9.):

macro_dde	9, B/I	MACRO_function: String; Senden eines (oder mehrerer, durch Blank getrennte) MACRO-Befehles an das picCOLOR-Programm zur sofortigen Ausführung. Dies kann unabhängig vom übrigen MACRO-Programm aus geschehen. Bei eingeschaltetem DDE-Transfer werden die Rückgabewerte unter dem Item ‘‘MResult’’ per DDE zur Verfügung gestellt. Das ist auch der Sinn dieser Funktion, daß hiermit alle Returnwerte über DDE verschickt werden können. (Funktioniert auch für TCP/IP). Wenn die Funktion mit eingeschaltetem "verbose"-Flag (Debug) ausgeführt wird, so werden alle gesendeten Befehle ins Notebook übertragen. Der Return-Wert Block beginnt mit: *80* ReturnValue = 1 ... usw...
macro_vars	1, B/ ID	inter: integer; Besetzen der Macro-Variablen IVAR[10]..IVAR[19], FVAR[10]..FVAR[19] und STRING[0](das ist die gesendete Funktion bei macro_dde) und STRING[1]. inter = 1: Anzeige in einer Dialogbox, inclusive des Return_Values aus IVAR[0] inter = 0: Ausgabe auf den System-Datenfile, wenn dieser geöffnet ist, ebenfalls inclusive des Return_Values aus IVAR[0]des letzten Funktionsaufrufes. Der Return-Wert Block bei der Fileausgabe beginnt mit: *80* ReturnValue = 1 IVAR[10] = ... usw... inter = -1: Nur Variablen-Rückgabe.
macro_control	2, B	single_step, show_only: integer; Umschalten in den Test-Modus während eines Macro-Ablaufes. single_step: -1 | 0 | 1 (-1: don't change) show_only: -1 | 0 | 1 (-1: don't change)
msg_interactive	1, B,	interactive_delay: integer; Umschalten aller Funktionen, die eine Message ausgeben: Bei Interactive>=0 verschwindet die Message nach kurzer Zeit (delay in Sekunden) automatisch, bei Interactive=-1 muß sie durch den Benutzer quittiert werden. Nach Beendigung des Macros wird diese Funktion immer automatisch auf -1 gesetzt. Return_Value ist immer 1. Wenn die Message automatisch verschwindet, wird intern angenommen, daß sie mit dem OK-Schalter bestätigt wurde. Dies ist wichtig bei Meldungen, die zwei oder drei Auswahlmöglichkeiten bereitstellen oder die mit linker oder rechter Maustaste oder mit der ESC-Taste gewählt werden können.
stop_on_error	1, B,	contin_stop_warn: integer; Error-Behandlung für das MACRO-Programm. Üblicherweise (contin_stop_warn=1) führt ein fataler Fehler, wenn eine Funktion als ReturnValue eine -1 liefert, sofort zum Fehlerabbruch. Eine ReturnValue von 0 ergibt eine Warnung. Wenn "contin_stop_warn" auf 0 gesetzt wird, erfolgt kein Fehlerabbruch mehr und es werden auch keine Warnungen ausgegeben. Bei "contin_stop_warn"=2 werden bei fatalem Fehler und bei einfachem Fehler nur Warnungen ausgegeben. Auf diese Weise können Fehler vom Benutzer abgefangen und weiterbehandelt werden. Nach Beendigung des Macros wird diese Funktion immer automatisch auf 1 gesetzt. Return_Value ist 1, außer wenn der Parameter unzulässig ist.
user_interaction	2, B	interaction_off_on, inhibit_paint: integer; In der Regel ist während der Ausführung eines MACRO-Programmes jede User-Interaktion im picCOLOR-Programm außer bei interaktiven Funktionen oder eines Abbruchs mit der ESC-Taste gesperrt. Mit interaction==1 kann diese Sperre aufgehoben werden. In diesem Fall muß man sehr vorsichtig sein, um nicht das MACRO-Environment zu verändern oder zu zerstören. Wenn das MACRO z. B. auf einem aktiven Bildspeicher x arbeitet und man aktiviert interaktiv duch Mausklick einen anderen Bildspeicher y, so wird die nächste MACRO-Funktion auf diesem neu aktivierten Bildspeicher ablaufen, sicherlich in der Regel ein unerwünschter Effekt. Zur Sicherheit kann im User-Menü über die Markierung des Punktes ‘‘MACRO Running...’’ überprüft werden, ob gerade ein MACRO abläuft, daß besser nicht gestört werden sollte, oder ob das MACRO bereits beendet ist und eine Interaktion sicher durchgeführt werden kann. Eine Bedienung anderer Programme unter WINDOWS ist natürlich in jedem Falle während eines MACRO-Ablaufes möglich. Hierzu muß nur der Mauszeiger aus dem picCOLOR-Programmfenster herausbewegt werden und steht dann für andere Programme zur Verfügung.

		inhibit_paint: -1|0|1: Wenn zu 1 geschaltet, wird kein Bildschirm-Update mehr durchgeführt. -1 gilt für beide Parameter als "don't change".
execute_macro	1,B/I	interactive: integer: Aufruf eines MACRO-Programmes. Da Macro-Programme noch nicht reentrant sind und da ein Macro-Programm nicht ein neues Macro-Programm aufrufen darf, sollte die Funktion nur über die DDE- oder TCP/IP-Schnittstelle zum Aufruf eines Macro-Programmes benutzt werden. Wenn diese Funktion von einem Macro aus aufgerufen wird, kann für nichts garantiert werden... inter = 1: normale interaktive File-Eingabe inter = 0: benutze Data-Pathname/Filename inter = -1: benutze Notebook-Inhalt (oder I/O-Box)

Buffer Initialisierung (Kapitel 6.12.):

init_buffer	4, B	buftype, dx, dy, initroicross: integer; Definition der Bildspeichergröße. Ersetzt die ältere initbuffer()-Funktion. Neu sind die Variablen dx und dy, mit denen die Bildgröße pixelgenau (evtl. modulo 8 oder 16) vorgenommen werden kann, wenn buftype auf 0 gesetzt wird. Wenn buftype > 0 ist, werden wie bisher die vordefinierten Buffertypen benutzt. buftype = 0: dx und dy werden für die Bildspeichergröße benutzt. In x-Richtung kann hierbei eine Beschränkung auf modulo 8 oder 16 implementiert sein. In y-Richtung kann auch eine Beschränkung implementiert sein. Dies kann hardwareabhängig sein. Fest definierte Speichergrößen: buftype = 1: 512* 512 Pixel, 6: 1280*1024; 2: 768* 512, 7: 2048*1024; 3: 1024* 512; 8: 2048*2048; 4: 1024* 768; 9: 4096*4096; 5: 1024*1024; 10: Auf Wunsch spezielle, benutzerdefinierte Größe; Wichtig ist der Parameter iniroicross. Er sollte in der Regel zu "1" gesetzt werden, damit alle ROIs und Fadenkreuze auf die neue Bildspeichergröße justiert werden. Bei "0" werden die alten ROIs und Fadenkreuzpositionen beibehalten. Return_Value ist 1 bei einwandfreier Funktion, -1 im Fehlerfall, d. h. wenn nicht genug Speicher für die gewählte Konfiguration zur Verfügung steht oder wenn unzulässiger buftype eingegeben wurde.
define_imagebuf	1, IM,	interactive: integer; Definition eines neuen Bildbuffers. Return_Value ist 1 bei erfolgreicher Definition, -1 bei Fehler, d. h. wenn kein Speicher zur Verfügung steht oder wenn zu viele Bildbuffer definiert sind (z. Zt. max. 256). interactive = 0: nicht interaktiv interactive = 1: interaktiv über Dialogbox interactive = -1: nichtinteraktiv, keine Benutzung des Speichers als Acquisition-Speicher möglich (nur für Frame Grabber, die diese Möglichkeit bieten)
free_imagebuf	1, IM,	interactive: integer; Freigeben des letzten Bildbuffers. Return_Value 1 bei Freigeben des Buffers, -1, wenn andere Buffer zuerst freigegeben werden müssen (Color_Buffer, 3D_Buffer), auch -1, wenn dieser letzte Speicher gerade von einer FrameGraber-Karte in Anspruch genommen wird. (interactive not used)
define_colorbuf	1, IM,	interactive: integer; Definition eines Farbbildbuffers (RGB und, wenn erwünscht und genug Speicher vorhanden, auch ein zweiter Farbraumspeicher, z. B. für HSI oder HLS). Return_Value 1 bei Definition, -1 bei Speichermangel, 0, wenn es Probleme mit der Dialogbox gab.
free_colorbuf	1, IM,	interactive: integer; Freigeben des letzten Farbbildbuffers. Return_Value 1 bei Freigeben, -1, wenn kein Color-Buffer definiert ist oder wenn andere Buffer zuerst freigegeben werden müssen, auch -1, wenn dieser letzte Speicher gerade von einer FrameGraber-Karte in Anspruch genommen wird. (interactive not used)
def_3d_image	4, B	w,h,d,acq_buffer: integer; Definiere 3D/Sequenz-Speicher (siehe 3D-Funktionen) w = Breite in Pixeln, h = Höhe in Pixeln, d = Tiefe in Pixeln oder Sequenzlänge in Bildern acq_buffer = 0|1: Man hat die Wahl, die Möglichkeit zum Gebrauch als Acquisition-Speicher abzuschalten, falls der FrameGrabber diese Möglichkeit bietet. Return Value = 0, falls die allocierte Größe nicht der geforderten entspricht, weil z. B. schon ein 3D/Sequenz-Speicher existiert (alle müssen die gleiche Größe aufweisen). Wenn schon ein Bild definiert wurde, so haben die Parameter keine Wirkung: das neu definierte Bild hat gleiche Größe und Länge.
free_3d_image	0, B	kein Parameter; Abmelden des letzten Sequenz/3D-Bildes und Freigeben des Speichers. Return_Value 1 bei Freigeben, -1, wenn kein 3D/Sequenz-Buffer definiert ist oder wenn andere Buffer zuerst freigegeben werden müssen, auch -1, wenn dieser letzte Speicher gerade von einer FrameGraber-Karte in Anspruch genommen wird.

Bildschirm und Bildspeicher (Toolbox Kapitel 13. und Control Panel 6.12.):

transfer_buffer	2, B,	Source_Buffer, Destination_Buffer: integer; Kopieren eines Bildbuffers in einen anderen. Return_Value 1 bei einwandfreiem Transfer, -1 bei unzulässiger Bufferangabe.
transfer_to_undo	1, B (IM),	Bildbuffer: integer; Kopieren eines normalen Bildspeicherinhaltes in den Undo-Buffer und Setzen der Undo-Parameter (nur wirksam, wenn die Undo-Funktion eingeschaltet ist ‘‘enable_undo (1)’’. Wenn Bildbuffer==-1 gesetzt wird, so wird der aktive, d. h. der angezeigte Speicher übertragen. Alle Bildattribute (LUT, Schwellen und Color-Bar usw.) werden mit übertragen. Return_Value 1 bei einwandfreiem Transfer, 0, wenn ‘‘UNDO’’ abgeschaltet ist (z. B. mit der Funktion ‘‘enable_undo (0)’’), -1 bei unzulässiger Bufferangabe, wenn z. B. ein Echtfarb- oder 3D/Sequenzspeicher aktiv ist.
enable_undo	1, B,	Undo_off_on: integer; Einschalten oder Ausschalten der Undo-Funktion für normale Bildspeicher: Wenn der Effekt, d. h. die Bildverarbeitung des gesamten Macro-Programmes nach seiner Ausführung zurückgenommen werden soll, so muß die interne Undo-Funktion ausgeschaltet werden, da sonst nur die jeweils letzte bildverändernde Funktion zurückgenommen werden kann. Vorher muß das aktuelle Bild in den Undo-Buffer mit ‘‘transfer_to_undo (-1)’’ kopiert werden. Eine übliche Befehlsfolge sieht folgender-maßen aus: transfer_to_undo -1 enable_undo (0) ... ... (Macro-Befehle) ... enable_undo (1) Nach Beendigung des Macros wird Undo_off_on automatisch wieder auf den Wert zurückgesetzt, der bei Start des Macros aktiv war. Return_Value immer 1.
show_all_buffers	2, B,	img_3dseq, count: integer; Anzeige aller Bildbuffer in verkleinerter Form (danach bitte wieder einen Bildbuffer aktivieren mit ‘‘select active’’). Return_Value immer 1. Auch zur Darstellung der Bilder einer Sequenz oder der Schnitte eines 3D-Bildes: Bei img_3dseq=0 werden die normalen Grauwert-Bildspeicher angezeigt, bei img_3dseq=1 die Schnitte des aktuellen Sequenz/3D-Bildspeichers. count gibt an, ob jedes oder jedes wievielte Bild angezeigt werden soll (letzteres nur bei img_3dseq=1 gültig).
select_active	2, B / I,	interactive: integer; buffer: integer; Aktivierung eines Bildbuffers (nur vorhandene Buffer aktivieren!). Wenn ‘‘interactive=1’’ ist, hat der Parameter ‘‘buffer’’ keinerlei Einfluß. Es sollte dann eine 0 eingegeben werden. Return_Value im interaktiven Modus 1 bei zulässiger Bufferbezeichnung (0<=buffer<n), sonst -1. Im Batch-Modus 1 bei zulässigem definierten Buffer, sonst -1.
select_second	2, B / I,	interactive: integer; buffer: integer; Selektion eines Bildbuffers für duale Funktionen (nur vorhandene Buffer aktivieren!). Wenn ‘‘interactive=1’’ ist, hat der Parameter ‘‘buffer’’ keinerlei Einfluß. Es sollte dann eine 0 eingegeben werden. Return_Value im interaktiven Modus 1 bei zulässiger Bufferbezeichnung (0<=buffer<n). Im Batch-Modus 1 bei zulässigem definierten Buffer, sonst -1.
get_xy_max	0, B	kein Parameter; Übergabe der Buffergröße und der sichtbaren Bildschirmgröße: IVAR[10] = XMIN IVAR[12] = XMAX IVAR[14] = DX_visible IVAR[11] = YMIN IVAR[13] = YMAX IVAR[15] = DY_visible
get_buffer	0, B,	kein Parameter; Die Nummer verschiedener Bildspeicher wird übergeben: IVAR[10] = Active Image Buffer IVAR[11] = Second Image Buffer (oft auch Destination Buffer) IVAR[12] = Last Image Buffer (letzter definierter normaler Bildspeicher, 1..n) IVAR[13] = Active RGB Buffer IVAR[14] = Second (Destination) RGB Buffer IVAR[15] = Last RGB Buffer (letzter definierter RGB-Speicher, 1..n) IVAR[16] = Active 3D/Sequence Buffer IVAR[17] = Second (Destination) 3D/Sequence Buffer IVAR[18] = Last 3D/Sequence Buffer (letzter definierter 3D/Sequenz-Speicher, 1..n) Return_Value immer 1.
select_activergb	2, B/I	inter, rgb_buffer: integer; Auswahl und Aktivierung eines bereits definierten TrueColor-

		Bildspeichers. interactive = 0|1
set_RGB_active	2, B/I	interactive, buffer: integer; Auswählen des RGB-Bildspeichers, der von anderen Funktionen zum Kopieren oder Transferieren angesprochen wird. Das kann z. B. das einzelne Transferieren der Rot/Grün/Blau-Farbebenen in normale Bildspeicher zur Weiterverarbeitung sein. ReturnValue 1 bei definiertem TrueColor-Buffer, sonst -1.
get_color_buffer	1, B	colorplane: integer; get information on requested color plane: IVAR[10] = actual plane in buffer (0=Dither, 1|2|3=R|G|B, 3|4|5=Second Color Space IVAR[11] = useful LUTavailable IVAR[12] = Active True Color Image Buffer (1..n) IVAR[13] = Destination True Color Buffer (Second) (1..n) IVAR[14] = True Color Image Count (1..n)
select_active3d	2, B/I	inter, buf: integer; Auswahl und Anzeige eines 3D/Sequenz-Bildspeichers. inter = 1: interaktive Dialogbox inter = 0: nicht interaktive Funktion buf: ist der Sequenz-Bildspeicher [1..n]. Wenn ein nichtexistenter Speicher gewählt wird, wird -1 als ReturnValue zurückgegeben.
set_3d_active	1, B,	3d_buffer: integer; Setzen des aktiven 3d-Buffers, aus dem z. B. durch 2D Bildspeicherfunktionen aus gelesen oder geschrieben wird. (1 <= 3d_active <= 3d-image-count)
set_3d_destin	1, B,	3d-buffer: integer; Setzen des zweiten 3d-Buffers für duale Funktionen (1 <= 3d_destin <= 3d-image-count)
get_3d_image	0, B	kein Parameter; Get 3D-image definitions: IVAR[10] = 3d-width IVAR[11] = 3d-height IVAR[12] = 3d-depth IVAR[13] = 3d-pixelsize IVAR[14] = 3d-image-count IVAR[15] = 3d-active IVAR[16] = 3d-destin Wenn kein 3D/Sequenz-Bild definiert ist, dann ist IVAR[14]=0. Return_Value immer 1.

ROI (Region of Interest) Einstellung (Manual Kapitel 11.4.):

select_roi	2, B/I,	interactive: integer; ROI-Nummer: integer; Wahl der aktiven ROI [0..2], kann interaktiv oder direkt ausgeführt werden. Die Eingabe der ROI-Nummer hat im Interactive-Mode (interactive=1) keine Wirkung. Return_Value im interaktiven Modus immer 1, im Batch-Modus 1 bei zulässiger ROI (0<=ROI-Nummer<=2), sonst -1. Hier entsprechen die ROIs x,y,z den Ziffern 0,1,2.
roi_size	4, B,	width, height, x_left, y_top: integer; Größe und Lage der aktuellen ROI einstellen. Return_Value ist 0, wenn die gewünschte ROI nicht auf den Bildschirm paßt und daher abgeschnitten wurde. Return_Value ist -1, wenn die ROI nicht zulässig ist, d. h. z. B. an allen Seiten über den Bildschirm hinausragt oder wenn width oder height negativ sind. Sonst ist der Return_Value = 0, if ROI oversize, but at least one side is still within image buffer limits; = -1, if all sides of ROI out of image buffer limits. In this case the ROI is maximized.
set_roi_max	0, B,	kein Parameter; Die aktuelle ROI wird auf maximale Größe gesetzt. Return_Value immer 1.
get_roi	0, B	kein Parameter; liefert die aktuelle ROI sowie deren Koordinaten zurück: IVAR[10] = ROI [0..2] IVAR[11] = Width IVAR[12] = Height IVAR[13] = X1 (linke obere Ecke) IVAR[14] = Y1 (")
roi_edit	2, I,	roi_coords, use_lens, change_roi: integer; Interaktive Definition und Einstellung der aktiven ROI und ihrer Größe und Lage. Außerdem einfache Meßfunktion und Lupenfunktion. ROI-Koordinaten werden bei roi_coords=1 angezeigt. Die Lupenfunktion wird bei use_lens=1 enabled. change_roi: 0|1: 0: keine Änderung der ROI-Nr. und des aktuelle Bildspeichers möglich.
show_roi	1, B,	hide_show_roi: integer; Hiermit kann die normale Anzeige der ROI-Umrandung abgeschaltet werden. Mit hide_show = -1 wird getoggelt. Diese Einstellung gilt für jeden Bildspeicher getrennt. (3D/Sequence not implemented) Return_Value immer 1.
copy_roi_coords	1, B	roi: integer; Kopiere Koordinaten der aktuellen ROI in die angegebene ROI (0,1,2 entspricht dabei den ROI's x,y,z)
move_roi	2, B	dx, dy: integer; Verschiebe aktuelle ROI. Wenn die ROI-Koordinaten durch die Verschiebung über den Bildschirmrand hinausgehen würden, wird die Verschiebung nicht durchgeführt und es wird -1 zurückgegeben.
saveload_roi	1, I	save_load: integer; Save ROI-coordinates to disk or load ROI-coordinates from disk - input file name (automatic extension: .ROI)
back_rest_roi	1, B	backup_restore: integer; Save ROI's and active ROI-number for later use or restore ROI's and active ROI-number after changes, when it has been backup'ed before.

Input und Output Funktionen:

long_input	9, I,	Hilfs_Message: string; Eingabe einer Integer-Zahl in die Integer-Variable IVAR[10]. Return_Value 1 bei richtiger Eingabe, 0 bei leerer Eingabe.
float_input	9, I,	Hilfs_Message: string; Eingabe einer Floating-Point-Zahl in die Float-Variable FVAR[10]. Return_Value 1 bei richtiger Eingabe, 0 bei leerer Eingabe.
string_input	9, I,	Hilfs_Message: string; Eingabe eines Textstringes in die Stringvariable STRING[1]. Return_Value 1 bei sinnvollem String, 0 bei leerem String.
long_output	1, IM,	Integer_Zahl: integer; Ausgabe einer Integer-Variablen. Erwartet einen Formatstring in der Stringvariable STRING[1]. Er muß ‘‘C’’-Konventionen folgen, d. h. mit %d (integer) oder %x (hexadezimal) werden Integer-Zahlen dargestellt. Wenn STRING[1] leer ist (=0x00), werden die Zahlen mit dem Kommentar: ‘‘Integer-Number’’ ausgegeben. Achtung: Wenn STRING[1] nicht sinnvoll besetzt ist, kann Unsinn ausgegeben werden! Die Behandlung von STRING[1] wie bei "user_message". Return_Value immer 1.
float_output	5, IM,	Float_Zahl: float; Ausgabe einer Floating-Point-Variablen. Erwartet einen Formatstring in der Stringvariable STRING[1]. Er muß ‘‘C’’-Konventionen folgen, d. h. mit %f (float, 4 Nachkommastellen fest) oder %e (Exponential, 4 Nachkommastellen fest) werden Floatingpoint-Zahlen dargestellt. Weitere Formatierungsparameter sind noch nicht implementiert. Wenn STRING[1] leer ist (=0x00), werden die Zahlen mit dem Kommentar: ‘‘FloatingPoint-Number’’ ausgegeben. Achtung: Wenn STRING[1] nicht sinnvoll besetzt ist, kann Unsinn ausgegeben werden! Die Behandlung von STRING[1] wie bei "user_message". Return Value ist immer 1.
error_box	9, IM,	Error_Message: string; Ausgabe einer beliebigen Meldung auf dem Bildschirm, die vom Benutzer quittiert werden muß. Return_Value immer 1.
user_message	9, IM,	User_Message: string; Ausgabe einer beliebigen Meldung auf dem Bildschirm, entweder als Messagebox oder in der Statuszeile des aktiven Bildes. Nur bei Anklicken der Ok/ Left/Yes-Taste oder bei Statuszeilenausgabe bei Anklicken der linken Maustaste wird das Macro normal weiterausgeführt, bei Cancel/Right/No oder Anklicken der rechten Maustaste wird eine Warnung ausgegeben und bei Eingabe der ESC-Taste wird die Ausführung des Macros abgebrochen. Return_Value 1 bei Bestätigung der Meldung mit Ok/Left/Yes oder der linken Maustaste, 0 bei Cancel/Right/No oder rechter Maustaste, -1 bei Abbruch mit der ESC-Taste. Mit einem String, der mit "LREStrng..." oder "YNString..." beginnt (jeweils genau 11 Zeichen), dann werden entsprechende Buttons in der Messagebox dargestellt, nämlich "Left/Right" oder "Yes/No", defaultmäßig "Ok/ Cancel". Wenn der String mit ‘‘Continue...’’ beginnt (das sind zusammen 11 Zeichen mit den 3 Punkten!), wird der Text in der Statuszeile ausgegeben und kann mit einem Mausklick irgendwo im Bild beendet werden. Wenn in STRING[1] spezielle Worte enthalten sind, kann ein Icon in der Ausgabebox erscheinen. Definiert sind z. Zt.: "Info...", "Warning...", "Error...", die dann zu den entsprechenden Icons (Ausrufezeichen blau im Kreis, Ausrufezeichen gelb im Warndreieck, Ausrufezeichen rot im Stopp-Schild) führen. Wenn diese Funktion dafür benutzt werden soll, vom Benutzer Entscheidungen zu verlangen, kann der Macro-Abbruch zuvor mit der Funktion stop_on_error 0 verhindert werden. Der Return_Value in IVAR[0] kann dann untersucht werden. Mann sollte direkt danach wieder stop_on_error 1 einschalten.
user_decision	4, IM,	s1, s2, s3, s4: integer; Der Benutzer erhält eine Dialogbox mit 4 Buttons und zugehörigem Text, von denen er eine Möglichkeit auswählen kann. Die 4 Textzeilen werden mit den Strings STRING[s1], STRING[s2], usw. vorbesetzt. Leere Strings ergeben anwählbare Buttons ohne Textzeile. Wenn jedoch ein Wert <0 ist, wird der jeweilige Button ausgegraut und inst nicht anwählbar. Die angewählte Zeile wird in IVAR[10] = 1|2|3|4 zurückgegeben, bei Cancel erhält man 0 zurück.
status_message	9, B	Status_Message: string; Ausgabe einer Message in der Statuszeile des aktiven Bildspeicherfensters. Die Meldung wird in der Regel von der nächsten Bildverarbeitungsfunktion überschrieben.
/* get_keyboard	0, I, */	kein Parameter; Fragt ab, ob eine Taste des Keyboards gedrückt wurde. Der ASCII-Code der Taste wird in IVAR[10] zurückgegeben. Wenn keine Taste gedrückt worden war,

		wird der Wert -1 zurückgegeben. Noch nicht richtig implementiert, da viele Funktionen das Keyboard selbst abfragen und den erhaltenen Tastencode dann nicht weitergeben.
get_mouse_button	0, I,	kein Parameter; Die Stellung der Mouse-Buttons wird abgefragt und in IVAR[10] für LEFT-Button und in IVAR[11] für RIGHT-Button zurückgegeben. Hierbei entspricht der Wert: 0: Button nicht gedrückt 1: Button gedrückt 2: Button und SHIFT-Taste der Tastatur (= mittlere Maustaste) gedrückt 3: Button und ALT-Taste der Tastatur gedrückt 4: Button und SHIFT und ALT-Tasten der Tastatur gedrückt
open_data_file	4, B / I	save_load, interactive, user_header, systemdata: integer; Open Data File für Schreiben oder Lesen. save_load = 0: Neuen File zum Schreiben erstellen und öffnen save_load = 1: Vorhandenen File zum Lesen öffnen interactive = 0: Filename/Pfadname über Funktionen eingeben interactive = 1: Namen interaktiv eingeben user_header = 0: Es wird der Header "*01* Measurement Data File" geschrieben,wenn print_data_headers gesetzt ist user_header = 1: Header wird eingefragt, wenn gleichzeitig interactive=1 ist systemdata = 0: Öffnen eines beliebigen Datenfiles systemdata = 1: Öffnen des System-Measurement-Files, auf den dann automatisch alle Messwerte des Systemes geschrieben werden Die Funktion gibt das File-handle (file_id) in IVAR[10] an das Macroprogramm zurück
close_data_file	1, B / IM	file_id: integer; Schließen eines Datenfiles mit Angabe der file_id (file-handle) Return_Value 1 bei erfolgreichem Schließen der Datei, sonst -1.
file_params	1, B	attributes: integer; Setzen bestimmter Windows-File-Parameter, normalerweise alle zu Null gesetzt. Für sehr schnelles streaming write sollte z. B. no_buffering und sequential_scan gesetzt sein, d. h. 0x0030 = 48 dezimal: attributes & 0x000F: 0 | 1 | 2 = norm | temp | write_through attributes & 0x00F0: 0 | 0x10 | 0x20 = norm | no_buffering | sequential_scan
data_params	3, B,	Headers, DecPoint, Delimiter: integer; Parameter für die Darstellung von Daten beim Abspeichern auf File. Headers: 0|1: Schalter, der eingeschaltet bewirkt, daß für jedes Datentupel eine Überschrift mitgespeichert wird. DecPoint: 0|1: Mit DecPoint=0 werden die Dezimalpunkte als Punkte abgelegt. Bei DecPoint=1 werden statt dessen Kommas benutzt. Dieser Schalter gilt auch für die Darstellung auf dem Bildschirm. Delimiter: 0|1: Defaultmäßig sind die Delimiter zwischen den Daten als Blanks ausgeführt. Bei Delimiter=1 werden statt dessen TAB-Zeichen verwendet. Dieser Schalter ist noch nicht implementiert. Return_Value immer 1.
data_filename	9, B	filename: string; Eingabe des Filenamens für einen Datenfile (max 256 Zeichen)
data_pathname	9, B	pathname: string; Eingabe des Pfadnamens für einen Datenfile (max 256 Zeichen)
open_comm_port	4, B	inter, ser_par, port, systemdata: integer; Öffnen einer Rechnerschnittstelle (Communication Port), z. B. LPT1 oder COM1. inter = 0|1: batch oder interaktiver Modus ser_par = 0|1: serielle oder parallele Schnittstelle port = 1|2|3|4 usw. ist die Portnummer, z. B. COM1, COM2, oder LPT1, LPT2 systemdata = 0|1: bei 1 wird der Systemdatenfile der Schnittstelle zugeordnet, d. h. alle Datenausgaben, die normalerweise auf den Systemdatenfile ausgegeben werden, laufen nun über die geöffnete Schnittstelle. Return Value ist 1 bei Erfolg, -1 bei Error
close_comm_port	0, B	kein Parameter; Schließen des zuvor geöffneten Communication Port Return Value ist 1 bei Erfolg, -1 wenn kein Comm Port geöffnet war.
comm_params	4, B	baud, bits, parity_stop, flow: integer; Parameter für die serielle Schnittstelle (COM1..n) baud = 110|300|600|1200|2400|4800|9600|14400|19200|38400|...| bits = 5..8

		parity_stop: (parity<<8) + stopbits: Parity = 0|1|2|3|4 = No|Odd|Even|Mark|Space Stop = 0|1|2 = 1|1.5|2 flow = 0|1|2|3 = None | Xon/Xoff | Dsr/Cts | KeepOld (Setzen des Flow-Controls)
open_stdoutdde	0, B	kein Parameter; Öffnen des Standard Output-Pfades. Wenn DDE Transfer eingeschaltet ist, wird dann auf die DDE-Schnittstelle ausgegeben. Gilt auch für den System-I/O-File. Return Value ist 1 bei Erfolg, -1 bei Fehlern oder in der DEMO-Version
close_stdoutdde	0, B	kein Parameter; Schließen des zuvor geöffnetenStandard-Output-Pfades Return Value ist 1 bei Erfolg, -1 wenn der Pfad nicht geöffnet war.
set_writef_id	1, B	file_id : integer; Setzen eines File-Handles vor dem Schreiben auf einen Datenfile. file_id muß der vorher beim Öffnen des Datenfiles erhaltene Wert sein. Es können mehrere Files geöffnet sein und man kann mit dem Handle festlegen, auf welchen dieser Files geschrieben werden soll. Bereits geöffnete Standard-Streams können benutzt werden: 0 = stdin 1 = stdout 2 = stderr 3..n = regular file handle, returned by open_data_file. Achtung: bei WINDOWS NT sind die Standard-Streams anders als früher bei DOS definiert. Die alten Streams stdaux und stdprn existieren leider überhaupt nicht mehr und werden nun simuliert, indem ein LPT- oder COM-Port geöffnet wird und dieser dann dem Standard-Input oder -Output zugeordnet wird. Reguläre File-Handles beginnen daher nun mit 3. Das picCOLOR-Programm besitzt hierzu eine Textedit-Dialogbox, die für Eingaben (stdin) und Ausgaben (stdout und stderr) benutzt wird und deren In- und Output z. B. auch über eine DDE-Schnittstelle von anderen Programmen angesprochen werden kann. Mit dem Parameter (-1) kann die File_id auf "nicht gesetzt" zurückgesetzt werden.. In diesem Fall werden alle Datenschreibvorgänge mit der I/O-Box durchgeführt.
writef_crlf	0, B (IM)	kein Parameter; Schreibe Carriage Return und Line Feed auf den zuvor aktivierten Datenfile.
writef_blnk	0, B (IM)	kein Parameter; Schreibe zwei Blanks (Leerzeichen) auf den zuvor aktivierten Datenfile.
writef_long	1, B (IM)	int: integer; Schreibe Integer Variable ohne zusätzliche Blanks auf den zuvor aktivierten Datenfile.
writefln_long	1, B (IM)	int: integer; Schreibe Integer Variable ohne zusätzliche Blanks und anschließend ein CRLF auf den zuvor aktivierten File.
writef_float	5, B (IM)	flt: float; Schreibe Floating Point Variable ohne zusätzliche Blanks auf den zuvor aktivierten Datenfile. Es wird die Exponentialschreibweise benutzt; es wird auf 4 Nachkommastellen gerundet.
writefln_float	5, B (IM)	flt: float; Schreibe Floating Point Variable ohne zusätzliche Blanks und anschließend ein CRLF auf den zuvor aktivierten Datenfile. Ausgabe in Exp.-Schreibweise auf 4 Nachkommastellen gerundet.
writef_string	9, B (IM)	str: string; Schreibe String ohne zusätzliche Blanks auf den zuvor aktivierten Datenfile.
writefln_string	9, B (IM)	str: string; Schreibe String ohne zusätzliche Blanks und anschließend ein CRLF auf den zuvor aktivierten Datenfile.
writef_byte	1, B	byte: integer; schreibe 1 Byte auf Output
set_readf_id	1, B	file_id: integer; Setzen eines File-Handles vor dem Lesen von einem Datenfile. file_id muß der vorher beim Öffnen des Datenfiles erhaltene Wert sein. Wenn mehrere Files geöffnet sind, so kann hiermit bestimmt werden, von welchem dieser Files gelesen werden soll. Bereits geöffnete Standard-Streams können benutzt werden: 0 = stdin 1 = stdout 2 = stderr 3..n = regular file handle, returned by open_data_file. Achtung: bei WINDOWS NT sind die Standard-Streams anders als früher bei DOS definiert. Die Streams stdaux und stdprn existieren leider überhaupt nicht mehr und werden nun simuliert, indem ein LPT- oder COM-Port geöffnet wird und dieser dann dem

		Standard-Input oder -Output zugeordnet wird. Reguläre File-Handles beginnen daher mit 3. Das picCOLOR-Programm besitzt hierzu eine Textedit-Dialogbox, die für Eingaben (stdin) und Ausgaben (stdout und stderr) benutzt wird und deren In- und Output z. B. auch über eine DDE-Schnittstelle von anderen Programmen angesprochen werden kann. Mit dem Parameter (-1) kann die File_id auf "nicht gesetzt" zurückgesetzt werden. In diesem Fall werden alle Datenlesevorgänge mit der I/O-Box durchgeführt. (**Achtung:** Alle Inputfunktionen liefern jetzt den Eingabewert in den Macro-Variablen ab Index-Nummer 10 zurück. Interaktive Eingabe einer Integerzahl geschieht in IVAR[10], einer Floating Point-Zahl in FVAR[10]. Nur beim Input eines Strings wird wie bisher in STRING[1] zurückgegeben.)
readfln_long	1, B	count: integer; Lese "count" Integer Variable bis zum Zeilenende (CRLF) von dem zuvor aktivierten ASCII-Datenfile. Der Wert wird ab den Variablen IVAR[11] gespeichert. Es werden maximal 9 (mehr über maxnum Eingabe: s.u.) Werte der Zeile eingelesen (bis IVAR[19]). Die Anzahl der eingelesenen Werte wird in IVAR[10] zurückgegeben. Das Einlesen bricht beim ersten nicht numerischen Zeichen ab.
readfln_float	1, B	count: integer; Lese "count" Floating Point Variable bis zum CRLF von dem zuvor aktivierten ASCII-Datenfile. Der Wert wird ab den Variablen FVAR[11] gespeichert. Es werden maximal 9 (mehr über maxnum-Eingabe: s.u.) Werte der Zeile eingelesen (bis FVAR[19]). Die Anzahl der eingelesenen Werte wird in IVAR[10] zurückgegeben. Das Einlesen bricht beim ersten nicht numerischen Zeichen ab.
readfln_maxnum	1, B	maxnumbers: integer; Setzen der Maximalanzahl von Zahlen, die mit einem der Befehle readfln_long oder readfln_float pro Zeile gelesen werden können. Defaultmäßig sind dies 9 Werte, die in den Variablen IVAR[11]..[19] und FVAR[11]..[19] zurückgegeben werden. Hier kann ein anderer Wert eingetragen werden, und die Werte können dann auch die weiter folgenden IVAR[20]... und FVAR[20]... überschreiben. Also bitte mit Vorsicht benutzen! Der ursprüngliche Wert für maxnum wird in IVAR[10] zurückgegeben. Achtung: die maximale Stringlänge von derzeit 512 Characters kann beim Einlesen jedoch nicht überschritten werden. Bei einer üblichen Länge einer Floating-Point-Zahl von z. B. 15 Zeichen können also trotzdem nicht mehr als 34 Zahlen pro Zeile eingelesen werden. Der Return-Wert ist immer 1.
readfln_string	0, B	kein Parameter; Lese String Variable (mit allen Blanks bis zum Zeilenende: CRLF) von dem zuvor aktivierten ASCII-Datenfile. Der Wert wird auf die Variable STRING[1] geschrieben.
readf_byte	0, B	kein Parameter; Lese ein Byte vom zuvor geöffneten File und speichere es als Integer-Wert auf die Variable IVAR[10].

DOS Funktionen (Manual Kapitel 6.12.10.):

directory	0, ID,	kein Parameter; Nach interaktiver Eingabe eines Such-Stringes Anzeige des DOS-Directory. *.* zeigt z. B. das gesamte aktuelle Directory an. Return_Value 1 bei DIR-Anzeige, -1 bei Fehler (DIR not found), 0, wenn kein Text eingegeben wurde.
work_dir	1, B/IM,	interactive: integer; Der Pfadname des aktuellen Working-Directory wird in die Variable STRING[1] übertragen und bei interactive==1 auch angezeigt. Return_Value immer 1.
change_dir	9, B/I,	Pfad-Name: string; Mit der Eingabe eines neuen Pfades wird ein neues aktuelles Directory definiert. Wenn Pfad-Name leer ist (""), wird der Pfad interaktiv eingefragt. Return_Value 1 bei richtigem Directory-Wechsel, -1 bei Fehler, 0 bei leerer oder mit ESC abgebrochener interaktiver Eingabe.
make_dir	9, B/I,	Dir-Nname: string; Mit der Eingabe eines Directory-Namens wird ein neues Directory im aktuellen Verzeichnis erstellt. Auch die Eingabe eines kompletten Pfad- und Directory-Namens ist möglich. Wenn Dir-Name leer ist (""), wird der String interaktiv eingefragt. Return_Value 1 bei erfolgreicher Erstellung, -1 bei Fehler, 0 bei leerer oder mit ESC abgebrochener interaktiver Eingabe.
remove_dir	9, B/I,	Dir-Name: string; Nach Eingabe eines Directory-Namens wird dieses Directory gelöscht. Auch die Eingabe eines kompletten Pfad- und Directory-Namens ist möglich. Wenn Dir-Name leer ist (""), wird der String interaktiv eingefragt. Return_Value 1 bei erfolgreichem Löschen, -1 bei Fehler, 0 bei leerer oder mit ESC abgebrochener interaktiver Eingabe.
work_drive	1, B/IM,	interactive: integer; Der aktuelle Working-Drive wird in die Variable IVAR[10] übertragen (als Ziffer: 0=A, 1=B, 2=C,...) und bei interactive==1 auch als Laufwerks-Buchstabe angezeigt. Return_Value immer 1.
change_drive	2, B/I,	interactive, nr: integer; Der aktuelle Working-Drive kann geändert werden. Bei interactive==0 wird nr zum neuen aktuellen Drive gesetzt (dabei ist 0=A, 1=B, 2=C,...). Bei interactive==1 wird der Drive interaktiv eingefragt. Hierbei werden Laufwerksbuchstaben eingegeben. Return_Value 1 bei erfolgreichem Drive-Wechsel, -1 bei Fehler, 0 bei leerer Eingabe.
delfile	9, B/I,	Filename: string; Ein File kann durch Eingabe seines kompletten Namens (mit Extension) gelöscht werden. Wenn kein Pfad angegeben wird, wird im aktiven Directory gelöscht. Wenn Filename leer ist (""), wird der String interaktiv eingefragt. Return_Value 1 bei erfolgreichem Löschen des Files, -1 bei Fehler, 0 bei leerer oder mit ESC abgebrochener interaktiver Eingabe.
filetime	9, B	filename: string; Actualize file date.
call_dos	9, B/I,	DOS-Command: string; Senden einer DOS-Kommandozeile. Wenn DOS-Command leer ist (""), wird der String interaktiv eingefragt. Die Funktion funktioniert natürlich nur solange im Batch-Modus, wie der abgesandte DOS-Befehl keine Interaktion benötigt. Wenn z. B. aus einem Batch-Macro ein interaktiver Texteditor aufgerufen wird, so muß man diesen natürlich interaktiv bedienen und schließen, um das Macro-Program weiterarbeiten zu lassen. Unter WINDOWS NT wird ein DOS-Fenster geöffnet, das sich nach Beenden der DOS-Funktion automatisch schließt. ReturnValue ist -1 bei fehlerhafter Ausführung des DOS-Befehles.

Lade- und Speicherfunktionen (Manual Kapitel 6.3. und 6.4.):

image_save	1, ID,	index: integer; Dialogbox zum Abspeichern eines Bildes. index ist der Eintrag in die Extension-Tabelle, sodaß eine Default-Extension definiert werden kann. 1:tif, 2:eps, 3:ppm, 4:bmp, 5:jpg, 6:sr3, 7:raw. Bei -1 ist tif der Defaultwert. Return_Value immer 1.
image_load	1, ID,	index: integer; Dialogbox zum Laden eines Bildes. index ist der Eintrag in die Extension-Tabelle, sodaß eine Default-Extension definiert werden kann. 1:tif, 2:bmp, 3:gif, 4:jpg, 5:ppm, 6:b16, 7:sr3, 8:*.*. Bei -1 ist tif der Defaultwert. Return_Value immer 1.
save_pathname	9, B,	Pathname: string; Eingabe des Pfades zum Abspeichen von Bildern. Return_Value immer 1.
load_pathname	9, B,	Pathname: string; Eingabe des Pfades zum Laden von Bildern. Return_Value immer 1.
save_filename	9, B,	Filename: string; Eingabe des Filenamens zum Abspeichern von Bildern. Return_Value immer 1.
load_filename	9, B,	Filename: string; Eingabe des Filenamens zum Laden eines Bildes. Return_Value immer 1.
show_image_header	1, I	image_header: integer; image_header = 0: Anzeige des picCOLOR-Bildspeicher-Headers image_header = 1: Anzeige des BMP-Headers des letzen geladenen BMP-Bildes image_header = 2: Anzeige des TIFF-Headers des letzten geladenen TIFF-Bildes image_header = 3: Anzeige der EXIF-Daten des letzten TIFF oder JPEG-Bildes
save_tiff	1, B (IM),	TIFF_Class: integer; Abspeichern enes Bildes: TIFF_Class: 2: Graustufen(8Bit), 3: Palettenbild(8Bit), 4: TrueColor-Bild (24Bit). Return_Value 1 bei erfolgreichem Abspeichern, -1 bei Speichermangel oder wenn bei einer Farbbildabspeicherung kein Color-Buffer definiert ist.
save_compress	1, B,	Compress: integer; Komprimieren des Bildes beim Abspeichern im TIFF-Format, wenn ‘‘Compress>0’’. Z. Zt. nur PackBit-Komprimierung unterstützt mit ‘‘Compress=1’’. Return_Value immer 1.
save_notebook	1, B,	Save_Note: integer; Abspeichern des Notebook-Inhaltes beim Abspeichern eines Bildes im TIFF-Format. Return_Value immer 1.
load_tiff	0, B (IM),	kein Parameter; Laden eines Bildes im TIF-Format. Return_Value 1 bei erfolgreichem Laden, -1 bei Fehler oder bei User-Abbruch infolge unbekannter TIFF-Kommandos. IVAR[10] = width IVAR[11] = height IVAR[12] = Bits/Sample IVAR[13] = Samples/Pixel IVAR[14] = TiffClass (2=gray, 3=palette, 4=rgb)
save_jpeg	1, B,	gray_color: integer; Save JPEG-komprimiertes Bild in Graustufen (8 Bit) oder Farbe (24 Bit RGB) (if gray_color=-1, use gray/color defined by internal variable TIFF_Class)
set_jpeg_quality	1, B,	Quality: integer; Qualität des komprimierten JPEG-Bildes 10 < Quality < 100. Sinnvoller Wert ist 75. Unter 50 wird die Qualität sehr schlecht!
load_jpeg	0, B,	kein Parameter; Lade JPEG-komprimiertes Bild. IVAR[10] = width IVAR[11] = height IVAR[12] = Bits/Pixel (8/24)
save_bmp	1, B/I	bw_color: integer; Speichern im Windows-Bitmap (BMP) Format. Es werden 8 Bit-Bilder (bw_color=0) oder 24 Bit-Bilder (bw_color=1) gespeichert.
load_bmp	0, B/I	kein Parameter; Laden eines Bildes im BMP-Format. Es können Bilder mit 1,2,4,8 Bit Grauwert- und 16 oder 24 Bit Farbtiefe geladen werden.
save_ppm	1, B	gray_color: integer; Abspeichern eines Grauwert- oder Farbbildes im PPM-Format

		(Portable Pix Map), wird z. B. für späteres Umwandeln in MPEG benötigt. Z. Zt. wird immer im Farbformat (P6) abgespeichert. Bei Graubildern sind dann R=G=B.
load_ppm	0, B	kein Parameter; Laden eines Grauwert- oder Farbbildes im PPM-Format (Portable Pix Map). Es werden P1,P2,P3, P4, P5, P6 unterstützt.
save_eps	1, B	gray_color: integer; Abspeichern im Postscript Format ".EPS" von Graustufen (8 Bit) oder Farb-Bildern (Achtung: 24-Bit-Farbbilder noch nicht getestet). gray_color = -1: use gray or defined by internal variable TiffClass as set in Dialogbox
load_gif	0, B	kein Parameter; Lade Bild im GIF-Format. Z. Zt. wird das GIF87a-Format unterstützt. Auch das GIF89a Format wurde beinahe komplett implementiert, Animated GIFs werden mit der Maus durchgeklickt. Noch nicht implementiert: Graphic Extensions, Plain Text. Außerdem kann es hier und da noch zu Fehlern kommen. ReturnValue = -1 bei nicht erkanntem und konvertiertem File.
load_b16	2, B	saturate, shift: integer; Laden eines Bildes im B16-Format. Dies sind 16Bit Grauwertbilder; das Format wurde durch die Firma PCO definiert. Da picCOLOR in der Regel in der 8Bit-Version vorliegt, muß ein 8 Bit breiter Bereich des 16 Bit-Pixels ausgewählt werden. Das wird mit dem shift-Wert getan. Bei shift=4 werden die Bits 4..12 eingelesen. Mit saturate = 1 kann entschieden werden, daß Werte, die den Grauwert 255 überschreiten, auf 255 gesetzt werden.
save_raw	1, B	bw_color: integer; Abspeichern von Bildern im RAW-Format, d. h. formatfrei, z. Zt. noch völlig ohne Header zeilenweise
raw_file_params	4, B,	w, h, bps_shift, header: integer; Einstellen der Parameter zum Laden eines Bildes in beliebigen unkomprimierten Formaten. Es muß eingegeben werden: Breite, Höhe, Pixeltiefe (8, 16, 24, 32 Bit), der Verschiebewert für Bilder mit mehr als 8 Bit Grauwerttiefe sowie die Länge eines Bildkopfes (Header) in Bytes. Pixeltiefe und Verschiebewert werden durch shift-Operation im Parameter bps_shift zusammengefaßt (Pixeltiefe<<8 | shift) mit bps = BitPerSample = 8 | 16 | 24 | 32 und shift in [0..32] als Verschiebung des eingelesenen 8-Bit-Wortes im 16, 24, oder 32-Bit Originalwort. header = Länge des Kopfdatensatzes des Bildes in Bytes. Wenn die header-Größe negativ angegeben wird, so wird nicht saturiert, d. h. Werte > 255 werden als Modulo(255) behandelt. Return_Value immer 1.
load_raw	0, B,	kein Parameter; Laden eines Bildes in einem speziellen Datenformat. Dieses Datenformat muß mit der Funktion raw_file_params eingestellt werden. Return_Value 1 bei erfolgreichem Laden, -1 bei Fehler. IVAR[10] = width IVAR[11] = height IVAR[12] = Bits/Sample IVAR[13] = shift right bits if pixel size > 8 bit/pixel
save_sr3d	1, B,	image: integer; Speichern eines 3D-Bildes der SwissRanger 3D-Kamera. image = [0..D3D_depth)
load_sr3d	1, B,	image: integer; Hereinladen eines 3D-Bildes der SwissRanger 3D-Kamera. image = [0..D3D_depth)
image_to_ROI	1, B,	Load_into_ROI: integer; Laden des Bildes in die aktuelle ROI oder in die linke obere Ecke des Bildes. ReturnValue immer 1.
roi_fit_image	0, B	kein Parameter; Einstellung der aktuellen ROI auf Größe und Lage des letzten eingelesenen Bildes. Das gilt für folgende Bildtypen oder Einlesequellen: TIF, JPG, BMP, PPM, RAW, GIF, B16, TWAIN-Schnittstelle, Zwischenablage.
enable_scale	1, B,	Load_Scale: integer; Schalter, ob beim Laden eines TIFF- oder BMP-Bildes eine eventuell mit abgespeicherte Kalibrierung mitgeladen werden soll. Load_Scale = 1: Lade und überschreibe Kalibrierung (Scale-Faktors). Return_Value ist immer 1.
d3dseq_save	1, ID,	index: integer; Dialogbox zum Speichern eines 3D/Seq.-Bildes. index ist der Eintrag in die Extension-Tabelle, sodaß eine Default-Extension definiert werden kann. 1:tif, 2:eps, 3:ppm, 4:bmp, 5:jpg, 6:sr3, 7:raw. Bei -1 ist tif der Defaultwert. Return_Value immer 1.

d3dseq_param	4, B	type, prm1, prm2, prm3: integer; Parameter für das interaktive Laden oder Speichern von 3D-Bildern oder Sequenzen. Hiermit können die Parameter der beim Laden oder Speichern erscheinenden Dialogbox vorbesetzt werden: type = 1: prm1 = RootCount, prm2 = NumberCount (für automatische Namensbildung) type = 2: prm1 = FileCount, prm2 = FileStart, prm3 = BufferStart Wenn ein prm < 0 gesetzt wird, wird der alte Zustand der internen Variable beibehalten.
d3dseq_load	1, ID,	index: integer; Dialogbox zum Laden eines 3D/Seq.-Bildes. index ist der Eintrag in die Extension-Tabelle, sodaß eine Default-Extension definiert werden kann. 1:tif, 2:bmp, 3:gif, 4:jpg, 5:ppm, 6:b16, 7:sr3, 8:*.*. Bei -1 ist tif der Defaultwert. Return_Value 1.
rt_savestream	4, B/I	interactive, namecount, numbercount, numberstart: integer; Einlesen von Kamera und Abspeichern eines Video-Streams in Echtzeit auf Festplatte. Hierzu muß ein geeigneter Frame Grabber mit Kamera zur Verfügung stehen sowie eine schnelle Festplatte oder ein schnelles RAID-System. Es kann ein Time-Stamp mit dem Bild mitgespeichert werden, der mit get_pix(0,0) abgefragt werden kann. IVAR[10] ist dann das MSB des Zählers. (diese Funktion ist im Zusatzmodul Echtzeitverarbeitung enthalten) Die Returnwerte geben hauptsächlich Auskunft über das Echtzeitverhalten und die Triggerkonditionen. IVAR[10] = Anzahl der Bilder im Stream IVAR[11] = 0, IVAR[12] = 0 (not used) IVAR[13] = Trigger Condition IVAR[14] = Ringbuffer Max. IVAR[15] = Ringbuffer Skip_Crash IVAR[16] = Ringbuffer Delay IVAR[17] = End Condition IVAR[18] = End Trigger Character (ASCII) IVAR[19] = Frames Lost FVAR[19] =Timestamp Maximum
timestamp_3d	3, B	count, io_status, timestamp: integer; Beim Speichern eines Videostreams können laufende Bildnummer, ein Wert einer digitalen I/O-Schnittstelle und/oder ein Zeitstempel in den ersten 12 Bytes jedes Bildes mitgespeichert werden, wenn der jeweilige Parameter zu 1 gesetzt wird. Jeder der Werte wird mit 4 Bytes in der Intel-Weise (umgekehrt) gespeichert.
load_stream	3, B/I	start, length, display: integer; Laden eines picCOLOR-Stream-Files. Dies sind Video-Streams, die mit einem einfachen Header (siehe Handbuch) als Sequenzen von Rohdaten gespeichert wurden. start: Bildnummer im Stream, mit der beim Laden begonnen werden soll. length: Anzahl der Frames, die eingelesen werden sollen. display = 0|1: Beim Einlesen des Streams kann ein Display erfolgen. Da der Display immer schon mit dem 1. Bild des Streams begonnen wird, kann hiermit auch der komplette Stream angeschaut werden.
save_avi	0, I	kein Parameter; Speichern einer unkomprimierten AVI-Sequenz des aktuellen Sequenz-Bildspeichers, vom im Sequence-Player eingestellten Start bis Ende, d. h. "From", "To". Die Sequenz wird im 24Bit-Farbformat gespeichert, d. h. in der Regel um den Faktor 3 aufgebläht, die maximale File-Länge beträgt 2GB. Mit anderen Programmen, z. B. VirtualDub, kann dann eine Kompression erfolgen.

Farbtabellensteuerung (Manual Kapitel 13.2.):

set_single_lut	4, B,	Adresse, Rot, Grün, Blau: integer; Setzen eines Adresseintrages in der Look-up-Table. Beispiel - Setzen des Pixelwertes 127 auf Gelb: set_single_lut (127, 255, 255, 0). Return_Value immer 1.
trans_lut	1, B,	Clear_lut_start: integer; Transferieren der Software-LUT in die Hardware-LUT. Wenn nur transferiert werden soll, muß Clear_lut_start=0 sein; sonst wird zusätzlich der Shift_LUT_Counter zurückgesetzt, d. h. eine verschobene Farbtabelle wird nicht mehr als verschoben erkannt. Return_Value immer 1.
show_gray	0, B,	kein Parameter; Erzeugen eines Graukeils nur in der Hardware-Farbtabelle, nicht jedoch in der zum aktuellen Bild gehörenden Softwaretabelle. D. h. es wird nur ein Grauverlauf angezeigt, nicht jedoch intern zum Bild gespeichert. Die alte Farbtabelle des aktuellen Bildes bleibt erhalten und kann mit der Funktion ‘‘trans_lut (0)’’ wieder angezeigt werden. Return_Value immer 1.
autolut	1, IM,	Farbtabelle: integer; Einfache Form der Funktion ‘‘select_lut’’. Setzen einer bestimmten Farbtabelle: Farbtabelle: 1: Rot, 2: Grün, 3: Blau, 4: HUE, 5:Lightness(Intensity), 6:Saturation, 7:Dither-Tabelle, 8:RGB-332-Tabelle, 9: Obj.-Class, 10: YUV: U, 11: YUV: V, 12:Cyan, 13:Magenta, 14:Yellow, 15:Thermo1, 16:Thermo2. Return_Value immer 1.
select_lut	3, B / I,	Farbtabelle, Param, Show_Palette: integer; Die gewünschte ‘‘Farbtabelle’’ kann aus der folgenden Tabelle ausgewählt werden: -1: Die letzte zuvor mit dieser Funktion gewählte Farbtabellenbelegungsnummer wird in IVAR[10] zurückgegeben. 1: Grauverlauf 2: Frame-Grabbing-Tabelle 3: Correction-Tabelle 4: Gray-Steps (Anzahl durch ‘‘Param’’ gegeben) 5: RGB-332-Tabelle 6: RGB-Dither-Tabelle 7: Rainbow-Tabelle 8: reserviert (Shift_lut) 9: reserviert (Band_lut) 10: LUT_Pixelmask (durch ‘‘Param’’ als 8Bit-Binary gegeben) 11: reserviert (Bin-Band_lut) 12: reserviert (single_lut) 13: HUE-Farbtabelle 14: Rot-Tabelle 15: Grün-Tabelle 16: Blau-Tabelle 17: First16 18: VGA16 19: Thermography 1 (Magenta am Ende) 20: Thermography 2 (für IR-Bilder, Weiß am Ende) 21: U (I)-Anteil im YUV (YIQ) Farbformate (Magenta-Grün) 22: V (Q) Anteil im YUV (YIQ) Farbformat (Blau-Gelb) 23: Cyan-Verlauf 24: Magenta-Verlauf 25: Yellow-Verlauf (8,9,11,12 sind reservierte Werte für die alte picCOLOR V1.0 TMS34010-Karte) Wenn ‘‘Show_Palette=1’’ ist, wird eine kleine Farbpalette angezeigt, die vom Benutzer wieder entfernt werden kann. Return_Value ist 1, bei den reservierten Einstellungen -1.
lut_box_mini	2, I	lut_mode, color_wheel: integer; Anzeige der kleinen Look-Up-Table Dialogbox lut_mode = 1: Wahl eines Bandes in der LUT, das auch zyklisch über die 0/255-Grenze reichen kann. Die Farbgebung wird zuvor durch die Funktion "bandfarb" eingestellt. lut_mode = 2: Einstellung eines binären Schwellwertes lut_mode = 3: Wahl eines binären Bandes, das auch zyklisch über die 0/255-Grenze reichen kann. lut_mode = 4: Zyklisches Verschieben des LUT-Inhaltes lut_mode = 5: Einstellung einer von 256 verschiedenen Anzahl von Grauwerten. color_wheel = 1: Es wird ein Farbenkreis zur Anzeige benutzt, wenn dies der eingestellte

lut_mode zuläßt (nicht bei lut_mode=5)
Return_Value immer 1.
Zurückgabewerte sind wie folgt definiert:

lut_mode:	1	2	3	4	5
IVAR[10]	Bandstart	Binstart	Binstart	Lut-start	Lut-steps
IVAR[11]	Bandend	Binend	-	-	-
IVAR[12]	Bandwidth	Binwidth	-	-	-

lut_box_maxi 0, ID, kein Parameter; Hervorbringen der großen Farbtabellen-Dialogbox. ReturnValue 1.

save_load_lut 2, IB, saveload, interactive: integer: Abspeichern oder Hereinladen einer Look-Up-Tabelle.

Printeransteuerung (Manual Kapitel 6.10.):

Z. Zt. gibt es hier keine Funktionen. Alle Printerausgaben werden interaktiv über die Windows-Printerschnittstelle durchgeführt.

Kopier- und Verschiebefunktionen (Manual Kapitel 13.5.):

copyarea	0, I,	kein Parameter; Interaktives Kopieren des aktuellen ROI-Inhaltes an eine andere Stelle des gleichen Bildes (in Zukunft vielleicht auch eines anderen Bildes, wie es bei der alten picCOLOR-Software möglich war). Akzeptieren mit der linken Maustaste, Abbruch mit der rechten Maustaste. Return_Value immer 1.
transp_mode	1, B	mode: integer; Transparenz-Mode für "copyarea" (nur für spezielle Hardware implementiert, z. B. picCOLOR V2.17-Karte) mode = 0: transparency on result=0 mode = 1: transparency on source = color0 (background color) mode = 4: transparency on result = 0 mode = 5: transparency on destination = color0 Achtung: Texas Instruments sagt, daß die Modes 1,4,5 Probleme haben könnten...
set_move_mask	3, B(ID)	interactive, off_on, color: integer; Setzen einer Maske für bestimmte Grauwerte (Pixelwerte) beim Kopieren, Verschieben oder Löschen. Bei interactive==1 kann auch die Transparency gesetzt werden. (nur für picCOLOR V2.17-Karte implementiert) color = [0..255]: Setzen oder Löschen, abhängig von "off_on", eines Pixelwertes (Grauwertes) in der Move-Mask. Return_Value immer 1.
move_transp	1, B,	Move_Transparency_off_on: integer; Einstellung der ‘‘Transparency’’ während der Operationen mit der Move-Mask: Diese Operationen sind z. Zt.: copy_wmask, move_wmask, kill_wmask. Return_Value immer 1. (nicht implementiert)
copy	2, B,	Source_ROI, Destination_ROI: integer; Kopieren eines Bildausschnittes von Source_ROI im aktiven Bild nach Destination_ROI im Second-Source- oder Destination-Bildbuffer. Dieser muß vor Ausführung richtig gesetzt sein. Pixel-Operation-Code oder Move-Mask sind nicht wirksam - es wird immer im ‘‘Replace’’-Modus kopiert. Wenn die Source_ROI größer als die Destination_ROI ist, wird z. Zt. noch über den Rand der Destination_ROI hinaus kopiert. Return_Value immer 1.
move	2, B,	Source_ROI, Destination_ROI: integer; Wie copy_ROI, nur daß der Inhalt des Source-ROI gelöscht wird. Return_Value 1 bei Ausführung der Funktion, -1 bei User-Abbruch.
copy_wmask	2, B,	Source_ROI, Destination_ROI: integer; Kopieren eines Bildausschnittes von Source_ROI im aktiven Bild zum Destination_ROI im Second-Source- oder Destination-Bildbuffer mit Hilfe des Pixel-Operation-Codes und der Move-Mask. Die Destination_ROI begrenzt den Kopierbereich. Return_Value immer 1. (nicht impl.)
move_wmask	2, B,	Source_ROI, Destination_ROI: integer; wie copy_wmask, nur daß der Inhalt der Source_ROI mit der Move-Mask gelöscht wird. (siehe kill_wmask). Return_Value 1 bei Ausführung der Funktion, -1 bei User-Abbruch. (nicht implementiert)
copy_rgb	0, I,	kein Parameter; interaktives Kopieren von Bildausschnitten im 24 Bit RGB-Buffer. Hierzu wird das geditherte Echtfarbbild auf dem Bildschirm benutzt, die Kopie dann jedoch im RGB-Buffer ausgeführt. Return_Value 1, außer wenn kein Color-Buffer definiert ist, dann -1.
copy_num	2, B	dx, dy: integer; kopiere Bildausschnitt im gleichen Bild mit numerischem Input. Wenn das Bild dabei über den Rand hinausgeschoben wird, wird einfach abgeschnitten. Der ReturnValue ist immer 1.

Löschroutinen (Manual Kapitel 13.7.):

clear_buffer	1, I/B,	interactive: integer; Löschen des aktuellen Bildschirmes und Bildspeichers, bei interactive=1 nach Sicherheitsabfrage. Return_Value 1 bei Löschen, 0 bei User-Abbruch.
clear_roi	0, B,	kein Parameter; Löschen der aktuellen ROI. Return_Value immer 1.
clear_out_of_roi	0, B,	kein Parameter; Löschen des Bildes ausserhalb der aktuellen ROI. Return_Value 1.
clear_roiborder	1, B	gray: integer; Löschen/Setzen des ROI-Randes im Bild gray = 0..255: ROI-Rand wird auf diesen Grauwert gesetzt
kill_wmask	1, B,	col: integer; Löschen der aktuellen ROI mit der Move-Mask - es werden nur Pixelwerte gelöscht, die in der Move-Mask eingeschaltet sind. Die in der Move-Mask gesetzten Pixelwerte werden zur Farbe col gesetzt. Return_Value immer 1. (nicht impl.)
clear_rgb	1, B/IM,	interactive: integer; Löschen des 24 Bit RGB-Buffers, bei interactive=1 nach Sicherheitsabfrage. Return_Value 1 bei Löschen, -1, wenn kein Color-Buffer definiert ist.
clear_hls_hsv	1, B/IM,	interactive: integer; Löschen des 24 Bit HSI- oder HLS-Buffers, bei interactive=1 nach Sicherheitsabfrage. Return_Value 1 bei Löschen, -1, wenn kein Color-Buffer definiert ist.

Bildübernahmefunktionen (Manual Kapitel 6.5.):

acq_grabbertype	4, B	fgtype, adjust, ext, video_standard: integer; Einstellung des Frame Grabber Typs, bestimmter Kameras/Video-Einstellungen und Standards. fgtype = 0: Disable Frame Grabber fgtype = 1: Internal Frame Grabber (falls vorhanden) fgtype = 2: External Frame Grabber (falls vorhanden) fgtype = 3: 2 externe Frame Grabber (falls vorhanden) fgtype = 10: Select FireWire (IEEE1394a) as standard input fgtype = 20: Select TWAIN as standard input fgtype = 30: Select AviCap WINDOWS Driver as standard input Wenn fgtype einen nicht erlaubten Wert hat (unterstützte interne Frame Grabber werden dem System z. Zt. bei der Kompilation mitgeteilt, eventuelle externe über den PICCNT.CFG-Konfigurationsfile, z. Zt. allerdings nur für picCOLOR V2.17 Karte), wird -1 zurückgegeben. Alle anderen Parameter sind alte, nicht mehr gebräuchliche Parameter. Sie sollten bei Verwendung der Funktion mit picCOLOR V4.0 zu -1 gesetzt werden: adjust: 0|1: Bei adjust=1 wird bei picCOLOR V2.17 bei Wahl des Acquire-Menüs die Videofrequenz auf 50 Hz gestellt (bei picCOLOR V4.0 nicht verwendet). ext: 0|1: Wenn 2 externe Frame Grabber im System sind, kann der Aktive hiermit ausgewählt werden. Wenn man sich im Frame Grabber Menü befindet, so wird auch das aktuelle Bild des ausgewählten Frame Grabbers angezeigt. video_standard = 0: RS-170 (EIA) Standard, video_standard = 1: CCIR-Standard video_standard = 2: KPF1e/k-Full Frame (picCOLOR V2.17) video_standard = 3: KPF1u-Full Frame (picCOLOR V2.17) video_standard = 4: JAI CV-M30 High Speed (picCOLOR V2.17) video_standard = 5: Test, z. B. LineScan (picCOLOR V2.17 Karte) video_standard = 12..15: Spezialkameras bei CORECO F64P-Karte Die tatsächlich nach Ausführung der Funktion eingestellten Parameter werden in den Macro-Variablen zurückgegeben, dabei sind nur IVAR[10] und IVAR[16] für das normale picCOLOR-Programm gültig: IVAR[10] = Grabber Type [0..3, 10,20,30,...] IVAR[11] = Adjust Frequency [0|1] IVAR[12] = Active External Grabber [0|1] IVAR[13] = Video_Standard [0|1|2|3|4|12..15] IVAR[14] = JAI CV-M30 Double Scan [0|1] IVAR[15] = JAI CV-M30 Partial Scan [100|50|33] IVAR[16] = Acquisition_OK: Kein Grabber = 0; Ja, 1 Grabber = 1; Ja, Stereo mit 2 Grabbern = 2; Ja, Stereo DoubleFull CameraLink mit 4 Grabbern = 4.
setup_capture	0, B/IM,	kein Parameter; Initialisieren des Bildübernahme-Environments bei zuvor eingestelltem FrameGrabber oder Bildübernahmeschnittstelle; vor einer Bildübernahme auszuführen. ReturnValue 1 bei erfolgreicher Ausführung, 0 bei Fehler (keine Framegrabberkarte im System o. a.). Eventuell muß ein time_delay folgen, z. B. 500ms.
close_capture	0, B,	kein Parameter; nach Beenden einer Bildübernahme auszuführen. ReturnValue = 1.
acq_config	4, B/I	inter, type, prm1, prm2: integer: Konfiguration von FrameGrabber Karte und Kameras. **inter = 1: Dialogbox** für Kameraauswahl oder FrameGrabber-Konfiguration: **type = 0:** Kameraauswahl. Hierzu müssen z. B. für die DALSA-CORECO-Sapera-Software die Filenamen für bestimmte Kamera- und Videoformat-Konfigurationsfiles angegeben werden. Der Kamera-File (Endung .CCA) besitzt Angaben über das Kamera-Timing, der Videoformat-File (.CVI) Angaben über Bildfrequenz und Bildauflösung. Nach Eingabe oder Auswahl der Filenamen werden beide Files mit der Taste ‘‘Load Cam- und Vic-Files’’ geladen und installiert. Bei Euresys-Karten wird nur ein Kamera-File (.CAM) definiert. Er wird erst beim Öffnen der Acquisition-Dialogbox geladen und installiert. In dieser Kamera-Selektions-Dialogbox können auch Stereo/Split-Screen-Parameter gesetzt werden oder einer von mehreren FrameGrabbern ausgewählt werden. **type = 1:** Setzen einer CallBack-Funktion und weiterer Parameter zur Bildübernahme, z. B. Time-Stamp und Bilddarstellung. Die CallBack-Funktion wird bei Bildbeginn oder Bildende automatisch aufgerufen und kann dann bestimmte Aktionen ausführen. Z. Zt. ist ein Auslesen der tatsächlichen Bildnummer oder die Abfrage eines Zeitgebers als Zeitstempel (interner Windows-Zeitgeber oder auf einer zusätzlichen Einsteckkarte) implementiert. Die Bildnummer ist eine laufende Nummer, beginnend mit 1. Wenn aufgrund Zeitmangels oder Rechnerüberlastung beim Bildeinlesen ein Bild verloren

gehen sollte, so kann dies an einer nicht fortlaufenden Bildnummer erkannt werden. Beim Zeitstempel kann ebenfalls ein fehlendes Bild an einem größeren Sprung im Zeitstempel erkannt werden. Außerdem kann hier eingestellt werden, ob bei einer Live-Bildübernahme eine parallele Anzeige der Bilddaten durchgeführt werden soll und ob dies für jedes eingelesene Bild über eine Callback-Funktion geschehen soll oder gesteuert über einen Hardware-Timer mit einstellbarer Frequenz. Diese Einstellung ist z. B. dann sinnvoll, wenn bei einer langsamen Graphikkarte und extrem schneller Kamera sonst eine Überlastung entsteht.

inter = 0: nicht interaktive Parametereingabe:
type = 0: Kameraauswahl. Für die DALSA-CORECO-Sapera-Software werden die Filenamen für bestimmte Kamera- und Videoformat-Konfigurationsfiles angegeben. Dies geschieht über die MACRO-Stringvariablen STRING[parm1] und STRING[parm2]. In STRING[parm1] wird die Kamera-File (Endung: .CCA), in STRING[parm2] der Video-File (Endung: .CVI) angegeben. Diese Strings müssen vor Ausführung der Funktion gesetzt werden. Bei Euresys-Karten wird nur der Kamera-File (.CAM) von STRING[parm1] geladen.
type = 1: Die Xfer-CallBack-Funktion wird über den Parameter parm1 ein- oder ausgeschaltet (1|0). Diese Xfer-Callback-Funktion ist die getestete Standard-Callback-Funktion bei der DALSA-CORECO- oder Euresys- oder IDS-Software. Diese Callback-Funktion wird nach jedem Transfer eines Bildes von der Acquisition-Einheit der Karte in den Bildspeicher aufgerufen.
type = 2: Die Acq-CallBack-Funktion (Nur DALSA-CORECO-Sapera) wird über den Parameter parm1 ein- oder ausgeschaltet (1|0). Diese Acq-Callback-Funktion wird bei jedem vertikalen Synchronsignal der Videoquelle aufgerufen, unabhängig davon, ob ein Bildtransfer stattfindet oder nicht. Diese Funktion ist nicht ausgiebig getestet.
type = 3: Anzeige-Modi der eingelesenen Bilder für Sequenzen. Mit parm1 = (0|1|2|3) kann die Anzeige abgeschaltet werden ‘‘0’’, oder es wird immer das erste Bild der Sequenz über eine Callback-Funktion angezeigt ‘‘1’’, oder es wird live mittels einstellbarem Hardware-Timer angezeigt ‘‘2’’, oder es wird jedes Bild über eine Callback-Funktion angezeigt ‘‘3’’. Bei der tatsächlichen Sequenzaufnahme wird ein eventuell eingeschalteter Timer (parm1==2) möglicherweise wieder abgeschaltet, um die Systembelastung für Echtzeitanwendungen zu minimieren.
type = 4: Anzeige-Modi der eingelesenen Bilder für normale Live-Acquisition in einen Bildspeicher. Mit parm1 = (0|1|2) kann die Anzeige abgeschaltet werden ‘‘0’’, jedes Bild über eine Callback-Funktion angezeigt werden ‘‘1’’, oder mittels Hardware-Timer mit einstellbarer Frequenz angezeigt werden ‘‘2’’.
type = 5: Setze High Resolution Timer als Time Stamp Source: parm1 = 0|1, oder setze Hardware-Time-Stamp (nur bei DALSA CORECO X64, DALSA-Xcelera, und bei Euresys GrabLink Karten) mit prm2 = 0|1. Achtung: nur einer davon darf gesetzt sein!
Rückgabeparameter:
IVAR[10] = TimebaseUnit für WINDOWS HighResTimer
IVAR[11] = TimebaseMax
IVAR[12] = TimebaseUnit Grabber
IVAR[13] =Timebase Max Grabber
Hierbei gilt: (1000=ms, 1000000=us). Per Default ist die Timebase der Frame Grabber in [us] eingestellt, des HighResTimers ebenfalls auf [us]!
type = 6: Setze External Timer als Time Stamp Source: parm1 = 0|1 (Hat Vorrang vor High Resolution Timer, z. Zt. bitte nur eine Time Stamp Source setzen!)
Ist keine Time Stamp Source gesetzt, so wird die einfache laufende Nummer benutzt. Der HighResTimer Wert ist in 1 Microsekunden-Schritten definiert. Der externe Timer/ Counter ist als Source noch nicht ausgiebig getestet. Er funktioniert bei Keithley-Timer-Karten nicht, bei den DII-Timer-Karten scheint er zu funktionieren.
type = 7: prm1 = WaitTimeOut[ms], prm2 = CommunicationTimeOut (nur DALSA)
type = 8: prm1 = 0|1|2 = normal single frame | split screen stereo | two screen stereo
type = 9: prm1 = 0: Active Grabber = 0; Second Grabber = 1, if possible
prm1 = 1: Active Grabber = 1, if possible; Second Grabber = 0. If DoubleFull-CameraLink is enabled, Active Grabber can be set to 0|1|2|3
type = 10: prm1 = 0|1: Set CalcFrameRate,
prm2: use internal buffers on frame grabber for temporary storage (0 oder 2..n, for DALSA-CORECO Sapera only) . Das kann bei PCI- oder PCIeBus-Flaschenhals sehr hilfreich sein!
type = 11: prm1 = 1: Reset FrameIndex: das nächste übernommene Bild erhält den FrameIndex 0
prm1 = 2: Reset FrameIndex to Default = Actual Image # in Ring Buffer
prm1 = 0: no change

		prm2: not implemented, for future reset of time stamp, use prm2=-1! **type = 12:** prm1 = 1: Set DoubleFull CameraLink with 2 grabbers per camera Für alle Funktionen gilt: parameters < 0 => don't change!
acquire	3, B/ ID,	interactive, type, norm_split_stereo: integer; Bildübernahmfunktion, für den aktiven FrameGrabber, auch für die FireWire-, TWAIN-, AviCap-Schnittstelle geeignet, wenn fgtype=10/20/30 ist (s.o.). interactive = 1: Es erscheint eine Dialogbox, in der alle Parameter gesetzt und die Bildübernahme gestartet werden kann. Wenn ein 3D/Sequenzbild aktiv ist, erscheint eine Videorekorder-Dialogbox. interactive = 0: Batch-Betrieb: type = -1: Acquire Live - kontinuierliche Bildübernahme. Wenn ein Sequenzbildspeicher aktiv ist und interactive == -1 ist, wird kontinuierlich durch den Sequenzspeicher hindurch übernommen, sonst wird nur live im ersten Bild der Sequenz übernommen. (Sehr Hardwareabhängig) type = 0: Freeze - Stoppen der Bildübernahme type = 1: Snap Shot - Einzelbildübernahme type = 2..n: Übernahme einer bestimmten Anzahl von Bildern (hardwareabhängig) type < -1: Sequenzbildübernahme, eventuell bei Hardwareeignung Übernahme einer bestimmten Anzahl von Bildern einer Sequenz: Absolutwert(type). norm_split_stereo: 0|1|2: Bildübernahme gleichzeitig von einer oder von zwei Kameras 0: eine Kamera 1: zwei Kameras im Split-Screen-Verfahren, wenn dies die Hardware erlaubt. Ob horizontal oder vertikal gesplittet werden kann, ist ebenfalls hardwareabhängig. 2: Stereobildübernahme von zwei Kameras in getrennte Bildspeicher. Hierzu müssen zwei Frame Grabber oder ein Frame Grabber mit zwei Eingängen vorhanden sein. ReturnValue normalerweise immer 1, -1 bei Fehler.
acq_snap_inc	3,B/ I	inter, type, norm_split_stereo: integer; Einzelbildacquisition im Sequenz- oder 3D-Bildspeicher. Hierzu muß zuvor set_singl_cont(0) gesetzt werden. Inter muß auf 0 und type auf 1 gesetzt werden! Sonst ist das Verhalten nicht definiert! Der erste Buffer einer Sequenz wird hier als Zwischenspeicher verwendet - in ihm befindet sich immer das letzte eingelesene Bild. Dies ist bei einigen Frame Grabbern aus hardwaretechnischen Gründen notwendig. Jeder Mausklick (oder die CR-Taste) liest ein Bild ein und inkrementiert den aktuellen Bildzeiger.
acq_average	3, B	count, avr_intgr, buf: integer; Bildmittelung durch Aufaddieren von ''count'' Bildern und späteres Dividieren durch ''count''. Funktion ist noch nicht implementiert. Sie kann jedoch simuliert werden, indem eine Sequenz der gewünschten Länge aufgenommen wird und danach die Average-Funktion aus dem Sequenz-Menü aufgerufen wird. Je nach Hardware kann diese Funktion auf Wunsch implementiert werden
acq_input_lut	2, B,	type, mask: integer; Setzen der Input-Look-up-table, z. Zt. nur bei der picCOLOR V2.17 Karte, bei der F64P-Karte und bei den DALSA/CORECO X64- und Xcelera-Karten: type =0: normaler Grauverlauf (0=schwarz, 1=weiß), type =1: invertierter Verlauf zur Übernahme von Negativen. mask: Bit-Maske zum Setzen oder Löschen der Übernahme von einzelnen Bits jedes Pixels. 1: Bit wird übernommen, 0: Bit wird zu Null gesetzt. Die Maske kann 8 oder 16 Bit breit sein, je nach Hardwareversion. Der Wert muß als Dezimalzahl eingegeben werden, d. h. 65535 ist 11111111 als normale Maske für vollständige Bildübernahme. Mit einer Maske 65534 (=11111110) kann z. B. das unterste Bit ausgeblendet werden.
acq_ext_sync	2, B/IM,	interactive, off_on: integer; interaktives oder batchgesteuertes Ein- und Ausschalten der Fremdsynchronsiation bei der picCOLOR V2.17 Karte oder F64P oder der IDS-Eagle. ReturnValue immer 1.
acq_cam_sync_inp	2, B,	camera_input, sync_input: integer; Auswahl des Kameraeinganges [0..3] und des Synchronisationseinganges [0..3] bei der picCOLOR V2.17-Karte. Bei der DALSA-F64P-Karte ist Sync-Input gleich Kamera-Input, außer wenn der Sync-Input auf 4 gestellt wird. Dann wird der CSI-Input benutzt. Bei Eingabe von -1 wird der Kamera- oder Sync-Eingang nicht geändert. Bei den IDS-Karten (Falcon, Eagle) sind Kamera- und Synchronsignal-eingang immer gleich. Wenn mehrere FrameGrabber eingebaut sind oder wenn eine Karte mehrere Akquisitionseinheiten besitzt, so werden diese mit acq_config umgeschaltet. ReturnValue 1 bei erfolgreicher Umschaltung, -1 bei unzulässigem Input.
acq_gain_offset	2, B,	contrast, brightness: integer; Einstellung von Kontrast [0..255] und Helligkeit[0..255].

		Normalwerte (default) sind: Kontrast = 0, Helligkeit = 127.
acq_editregs	0, ID	kein Parameter; Editiere FrameGrabber Register, nur für DALSA-CORECO F64P-Karte.
acq_square_pix	3, B (I),	interactive, type, freq: integer; Einstellung quadratischer Pixel bei Bildübernahme mit picCOLOR V2.17 oder DALSA-F64P-Karte. Es wird die Pixel-Frequenz in Hz eingefragt. Defaulteinstellung beträgt 14375000 (bzw 14750000 Hz bei F64P). Die Bildfrequenz (z.B. 50Hz und 15.625kHz) bleibt erhalten. ReturnValue immer 1. type = -1: reset frequency to 14375000 Hz type = 0: use "freq" mit 10000000<freq<16000000 type = 1: Hitachi KPM1ek mit 14187500 Hz type = 2: Hitachi KP503G mit 14750000 Hz Andere Werte sind bereits implementiert, bitte anfragen! Default ist 14375000 Hz. Übliche Analog-Videokameras liefern quadratische Pixel mit 14750000 Hz.
acq_conf_1394	4, B/I	inter, type, prm1, prm2: integer; Einstellung aller Parameter für die FireWire: inter = 1: type = 1 | 2 | 3 | 4: 1:Init-Dialogbox, 2:Acqisition-Dialogbox, 3/4: Partial-Scan-Dialogbox für die erste oder die zweite Kamera (falls Stereo eingeschaltet ist). inter = 0: type = 0: Wenn prm1 = 1, setze Grabbertype=10 für FireWire, sonst =1 type = 1: Check FireWire Link type = 2: Init Camera type = 3: Show Camera Model, type = 4: Show Max Speed, type = 5: select camera (prm1=first, prm2=second), type = 6: toggle external trigger (prm1=first, prm2=second cam), type = 7: toggle stereo mode, type = 8: measure frame rate (prm1=first, prm2=second), type = 9: set format and mode for first camera, type = 10: set format and mode for second camera (prm1=format=0|1|2, prm2=mode=0|1|2|3|4|5|6|7), type = 11: set rate for both cameras: prm1=2|4|7|15|30|60 type = 12: prm1 = XferCallBack = 0|1 type = 13: prm1 = Use HighresTimer = 0|1 (Return Base und Max) type = 14: prm1 = UseExternalTimer = 0|1 type = 15: prm1 = 1: Reset FrameIndex
acq_conf_twain	4, B/I	inter, type, prm1, prm2: integer; Einstellung der Parameter für die TWAIN-Schnittstelle: inter = 0: type = 13: prm1 = Use HighresTimer = 0|1 (Return Base und Max) type = 14: prm1 = UseExternalTimer = 0|1 type = 15: prm1 = 1: Reset FrameIndex
acq_conf_avicap	4, B/I	inter, type, prm1, prm2: integer; Einstellung der Parameter für die AviCap-Schnittstelle: inter = 1: type = 0 | 1 | 2 | 3 : Dialogbox inter = 0: type = 0: not implemented. type = 1: prm1 = AviCapCallBack = 0|1 type = 2: not implemented. type = 3: Anzeige-Modi für Sequenzen 0 | 1 | 2 | 3 type = 4: Anzeige-Modi für normale Bilder 0 | 1 | 2 | 3 type = 5: prm1 = Use HighresTimer = 0|1 (Return Base und Max) type = 6: prm1 = UseExternalTimer = 0|1 type = 7: not implemented. type = 8: 0|1|2 = normal single frame | split screen | two screen stereo type = 9: prm1 = 0: Active Grabber = 0; Second Grabber = 1, if possible prm1 = 1: Active Grabber = 1, if possible; Second Grabber = 0 type = 10: prm1 = 1: Reset FrameIndex
acq_conf_sr3d	4, B/I	inter, type, prm1, prm2: integer; Parameter für die SwissRanger 3D-Time-Of-Flight Kamera (TOF): inter = 1: type = 0: Acquire-Dialogbox type = 1: Config-Dialogbox type = 2: Init Interface type = 3: Exit Interface type = 4: Check DLL type = 5: Reallocate Buffers (can be 2D or 3D buffers, depends on active image buffer)

		type = 6: Live Configuration inter = 0: type = 1: prm1 = distance-offset -128..128 type = 2: prm1 = modulation type =3: Channel 0: prm1 = min, prm2 = max. Wenn beide <0, return min,max in IVAR[10] und IVAR[11] type =4: Channel 1: prm1 = min, prm2 = max. Wenn beide <0, return min,max in IVAR[10] und IVAR[11] type = 13: Setze High Resolution Timer als Time Stamp Source: parm1 = 0|1, siehe acq_config() type = 14: Setze External Timer als Time Stamp Source: parm1 = 0|1, siehe acq_config() type = 15: Setze Referenz-Frame-Index, siehe acq_config()
acq_1394	3, I	interactive, type, norm_split_stereo: integer; Interaktives Scannen von der 1394-Schnittstelle unter Umgehung der 1394-Aktivierung mit acq_grabbertype.
acq_twain	3, I	interactive, type, norm_split_stereo: integer; Interaktives Scannen von der TWAIN-Schnittstelle unter Umgehung der TWAIN-Aktivierung mit acq_grabbertype: type = -3: Select Source type = -2: Set Source type = -1: interactive Acquire Box. IVAR[10] = width, IVAR[11] = height type = 0: Batch TWAIN-Scanning, not yet implemented...
acq_avicap	3, I	interactive, type, norm_split_stereo: integer; Interaktives Scannen von der AviCap-Schnittstelle unter Umgehung der AviCap-Aktivierung mit acq_grabbertype.
acq_sr3d	1, I	type: integer; Interaktives Scannen von der SR3D-Schnittstelle unter Umgehung der SR3D-Aktivierung mit acq_grabbertype.
acq_wait_blank	0, B	kein Parameter; Wait for next blanking
acq_wait_odd	0, B	kein Parameter; Wait for next odd frame
acq_wait_even	0, B	kein Parameter; Wait for next even frame
set_singl_cont	1, B	single_continuous: integer; Kontinuierliche Aufnahme oder Einzelbildsteuerung für die normale acquire-Sequenzaufnahmefunktion.
set_oddfirst	1, B	even_odd_first: integer; Normalerweise startet die Bildübernahme beim ODD-Halbbild. Für spezielle Anwendungen kann hiermit die Sequenzübernahme mit dem EVEN-Halbbild begonnen werden. (nicht implementiert, hardwareabhängig)
set_interleave	1, B	interleave: integer; Interleave (Delay) for sequence grabbing: max frequency: interleave=1: 25 full pictures / second (bei CCIR) max interleave = 90000: 1 picture / hour (bei CCIR) (nicht implementiert, hardwareabhängig, kann auf Wunsch implementiert werden)
set_ringbuf	2, B	ringbuffer, checkgrabber: integer; Ring-Speicher-Einstellung: ringbuffer=1: Sequence-Speicher wird als Ring-Speicher genutzt, die Sequenz-Bildübernahme stoppt beim zweiten Mausklick. Der Beginn einer Sequenz liegt dann natürlich nicht am Anfang des Sequenz-Speichers, sondern muß erst mit der Funktion "rotate_3d" dorthin geschoben werden. checkgrabber = 1: check ring cycle mode capability for frame grabber. If grabber cannot change the cycle mode during acquisition, one has to exit and then reenter acquisition after adjusting the required ring mode. In this case the return value is 0.

Obsolete Funktionen, nur für picCOLOR V2.17-Hardware (eingeschränkt auch für DALSA-F64P-Karte:

acq_vid_params	4, B	interlaced, pixelclock, ext_trigger, special: integer; Alte Funktion zum Setzen verschiedener Videoparameter. Mit special = 20 können hier ebenfalls die Kamera- und Videoformatfiles geladen werden, siehe acq_config(). Mit dieser Funktion kann ein Externer Hardware-Trigger gesetzt werden (ext_trigger==1). Alle Parameter werden bei ‘‘-1’’ als ‘‘no change’’ aufgefaßt. (Funktion nicht im Gebrauch)
camera_type	1, B	type: integer; Setzen des Kameratyps (nur für picCOLOR V2.17-Karte implementiert)

		type = 1: Schwarz/Weiß-Kamera an Kamera_Input [0,1,2,3]. type = 2: RGB-Kamera an den Kamera_Inputs 1:R, 2:G, 3:B. type = 3: Schwarz/Weiß-Kamera mit einzelnen Farbfiltern
shift_video	2, B,	horizontal, vertical: integer; 0<horizontal<n, -m<vertical<m, Verschiebe übernommenen Bildausschnitt des Videosignales durch Anpassen der Austastlücken. Vertikale Verschiebung nach oben ist bei picCOLOR V2.17 nur bei Sequenzübernahmen möglich. Für Standbilder müßte der Videospeicher auf 4MBytes ausgebaut werden. Bei der DALSA-F64P-Karte ist die Funktion nur wirksam, wenn die ROI nicht maximal groß ist. (nicht implementiert, nur picCOLOR V2.17 und F64P-Karte)
extgrab_copy	0, B (IM),	kein Parameter; Kopieren eines Bildes von einem externen Frame Grabber zum Bildschirm in die aktuelle ROI bei Grauwertbildern oder in die RGB-Farbspeicher bei 24-Bit-Farbbildern. Bei Grauwertbildern muß noch eine Übertragung in den Bildspeicher durchgeführt werden (z. B. mit screen_to_buffer). ReturnValue im Fehlerfall = -1, d. h. bei zu kleinem Zwischenspeicher, bei ungültigem Frame-Grabber-Bild, bei falscher Pixelgröße, bei Farbbild und nicht definiertem RGB-Buffer. (nicht implementiert, nur für picCOLOR V2.17-Karte mit zusätzlichen externen Grabbern)
extgrab_roi	4, B	w,h,xleft,ytop: integer; Definieren einer ROI für einen externen Frame Grabber (nicht implementiert, nur für picCOLOR V2.17-Karte mit zus. ext. Grabbern)
extgrab_clear	0, B,	kein Parameter; Lösche innerhalb der vorher gesetzten ROI im externen frame grabber. (Bei internem Grabber: =clear roi) (nicht implementiert, nur picCOLOR V2.17-Karte)

Geometrische Korrekturen (Manual Kapitel 7.1.):

zoom	3, B,	Source_ROI, Destination_ROI, interpol: integer; Zoomen eines Bildausschnittes von der Source-ROI in die Destination-ROI. interpol = 0: keine Interpolation interpol = 1: lineare Interpolation in der 2*2 Nachbarschaft interpol = 2: Interpolation mit kubischen Splines in der 4*4-Nachbarschaft. Return_Value immer 1.
zoom_fast	2, B	source_roi, dest_roi: integer; Sehr schnelles Zoomen um ganzzahlige Faktoren (1,2,3,4,5,...) von der Source-ROI in die Destination-ROI
mirror	1, B	horiz_vert: integer; Spiegeln um vertikale oder horizontale Achse. ReturnValue immer 1.
rotate_quad	0, B,	kein Parameter; Rotieren der aktuellen quadratischen ROI um 90 Grad im Uhrzeigersinn. Return_Value 1 bei erfolgreicher Drehung, -1 bei nicht quadratischer ROI.
rotate_rect	3, B	clockwise, swin, dwin: integer; 90-Grad-Rotation: Rotieren der Source_ROI um 90 Grad und Ablegen des Ergebnisses in die Destination_ROI. clockwise = 0|1 = counterclockwise | clockwise swin, dwin = Source ROI, Destination ROI Return_Value 1 bei erfolgreicher Drehung, -1 bei zu breiter Source-ROI (Breite > Bildbufferhöhe).
rotate_angle	8, B	sx, sy: integer; angle, interpol: float; Drehung des Bildausschnittes um den Winkel "angle" [-180.0..180.0]. Gleichzeitig kann auch eine Verschiebung (Translation) um sx/1000, sy/1000 (in Pixeln) angegeben werden. interpol = 0: keine Interpolation interpol = 1: lineare Interpolation in der 2*2 Nachbarschaft interpol = 2: Interpolation mit kubischen Splines in der 4*4-Nachbarschaft. interpol = -1: lineare Interpolation, schnelles Verfahren mit FIXPOINT Algorithmus statt des normalen floatingpoint Algorithmus. Ist besonders viel schneller auf älteren Prozessoren - daher bitte vor Benutzung austesten! Es kann bei extrem großen Bildern wegen zu geringer interner Nachkommastellenzahl zu leichten Ungenauigkeiten führen. Kann dann auch zu Overflow führen, dann werden bestimmte Bildteile nicht gedreht. Bitte testen oder aktuelle Werte erfragen! Außerhalb der ROI liegende Bereiche werden mitbenutzt. Außerhalb des Bildspeichers liegende Bereiche werden zu "0" angenommen.
unroll	4, B	dstbuf, dstroi, ipol<<16|repeat, refrad: integer; Abrollen eines Bildes um dessen Mittelpunkt. Achtung: die ROI muß quadratisch sein! dstbuf = 1..n, der Destinationbildspeicher dstroi = 0|1|2, die Destination ROI ipol<<16|repeat = (0|1|2 << 16) | Anzahl der Wiederholungen (Rotationen) refrad = Referenzradius, der nicht gestaucht oder gedehnt wird (1..100%, -1=100%).
distort_cos	4, B/I,	ipol, preview, limit100, nia_y_x: integer; interaktives (oder auch im Batchverfahren ablaufendes) Verzerren eines Bildausschnittes mit einer 1+cos-Funktion. Return_Value immer 1. preview = 0: Es wird nur ein (sehr schnelles) Cosinus-Fadenkreuz als preview benutzt. preview = 1: Man sieht die tatsächliche Verzerrung währende des Arbeitsvorganges (jedoch nicht mit Interpolation) ipol=1: Subpixelgenaue Interpolation limit100 = [0..100], setzt ein Verzerrlimit innerhalb der gesetzten ROI nia_x_y == 0: interactive nia_x_y != 0: batch mode: shift = (y<<16 & x) (ROI must be set already) Wenn Stereo-Bildbuffer und das Stereoflag gesetzt sind, wird die Funktion auf beide Stereobildhälften ausgeführt. Die Fadenkreuze können dann dazu benutzt werden, eine Verschiebung zwischen beiden Bildern zu berücksichtigen.
polydist	4, B/I	type, refimage, interpol, coeffs: integer; Polynomial Distortion (Warping): Entzerrung eines Bildausschnittes mit zweidimensionalen Polynomen zweiten oder höheren Grades. ipol=[0,1,2] für [no interpolation, linear, cubic spline] (siehe auch "rotate_angle"). Außerhalb der ROI liegende Bereiche werden zur Hintergrundfarbe gesetzt (z. Zt. 128). Es werden interaktiv mehrere Paßpunkte eingefragt, zuerst für das verzerrte Bild im

		aktuellen Bildspeicher, dann für das Referenzbild im "Second-Image-Buffer". Bei Eingabe überbestimmter Paßpunkte werden diese zur Ausgleichsrechnung verwandt. Maximale Eingabe ist z. Zt. 20. Bei unterbestimmten Paßpunkten kommt es zu unsinnigen Ergebnissen!Unter bestimmten Bedingungen zeigt die Funktion leider noch fehlerhafte Ergebnisse. Daher sollte man vor einer Benutzung ausgiebig testen, ob sie für die eingestellte Entzerrung geeignet ist! coeffs = 3..10 gibt die Anzahl der berücksichtigten Glieder der Polynome der folgenden Form an: $x = a_0 + a_1x' + a_2y' + a_3x'y' + a_4x'^2 + a_5y'^2 + a_6x'x'y' + a_7x'y'y' + a_8x'^3 + a_9y'^3$ Bei coeffs=3 werden a_3, a_4, a_5 und folgende zu Null gesetzt, bei coeffs=4 wird a_3 mit-benutzt, bei coeffs=5 wird zusätzlich a_4, bei coeffs=6 zusätzlich a_5 berücksichtigt, usw. Return_Value 1, außer bei singulärer Transformationsmatrix, dann -1. type = -1: interactive type = 0: reset type = 1: do warping type = 2: save coeffs type = 3: load coeffs refimage: -1: don’t change, 1..n image buffer interpol = 0|1|2 = no|linear|spline coeffs = 3..10
distort_saveload	2, B/I	save_load, interactive: integer; Laden oder Abspeichern der Polynom-Distortion-Parameter
splinedist	1, I,	ipol: integer; Entzerrung eines Bildausschnittes mit zweidimensionalen Splines dritten Grades (noch nicht vollständig implementiert). ipol wie bei "polydist". Return_Value 1 bei erfolgreicher Transformation, -1 bei Fehler.
trapdist	8, B	ipol, iter: integer; alpha, camdist: float; Trapezförmige geometrische Entzerrung bei bekanntem Aufnahmewinkel und -abstand. ipol = 0|1|2 für [keine Interpolation, linear, und Spline] iter ist die maximale Anzahl interner Iterationen. Die Entzerrungsgleichungen können nur über eine Iteration gelöst werden. Defaultmäßig ist hier ein Wert von 10 sinnvoll. Zu geringe Iterationsanzahl bewirkt ungenaue Werte am linken und rechten Bildrand. Es wird dann eine Warnung ausgegeben. alpha ist der Kamerawinkel relativ zur Bildnormalen in der ROI-Mitte, in horizontaler Richtung (x-Richtung des Bildschirms) gemessen. Winkel in y-Richtung sind nicht zugelassen. Gegebenenfalls muß das Bild zuvor gedreht werden. camdist ist der Abstand der Kamera zur ROI-Mitte in Pixel-Einheiten. ReturnValue ist immer 1.
camcal	2, B	interpol, background: integer; Durchführen einer Kamerakalibration (Bildentzerrung) anhand von Parametern, die mit dem TSAI-Lochkameramodell bestimmt worden sind. Dabei kann mit interpol=(0|1|2) die Art der Interpolation eingestellt werden (keine|linear|spline) und der Hintergrund-Grauwert angegeben werden, der bei Überschreitung der Bildspeichergrenzen angewandt wird.
camcal_params	8, B	type, parm: integer; val1, val2: float; Einstellen der Parameter für Kamerakalibrierung. type = -3 | -2: interaktive Dialogbox: -3: do not allow function to execute in image buffer. type = -1: Reset Parameters (kappa = 0.0) type = 0: Initialize Camera Parameters, (Brennweite f = val1, kappa = val2 not used), uses actual ROI for centering - make sure to have optical axis in center of ROI! type = 1: Get/Set Camera Parameters: parm = -1: get all parameters: FVAR[10]=cp.Ncx, FVAR[11]=cp.Nfx, FVAR[12],[13]=cp.dx,dy, FVAR[14],[15]=cp.dpx,dpy, FVAR[16],[17]=cp.Cx,Cy, FVAR[18]=cp.sx. parm = 1: val1 = sel (sensor elements, Ncx), val2 = pixel (pix count in FrameGrabber, Nfx) parm = 2: val1=mm/sel in x-, val2=mm/sel in y-direction of camera (dx,dy) Diese Funktion setzt automatisch auch die Werte x,y in mm/pixel des Frame Grabbers (s.u.; parm = 3). Falls diese unabhängig davon gesetzt werden sollen, muß das nachträglich geschehen! parm = 3: val1=mm/pixel in x, val2=mm/pixel in y of FrameGrabber (dpx,dpy) parm = 4: val1=x, val2=y: Z axis intercept of camera coordinate system (Cx,Cy) parm = 5: val1 = scale factor to compensate for error in dpx (cp.sx) type = 2: Get/Set Calibration Constants:

parm = -2: transfer camera position to internal campos variable
parm = -1: get all parameters: IVAR[10]=cc.ipol, FVAR[10]=cc.f,
FVAR[11]=cc.kappa1, FVAR[12],[13]=cc.p1,p2,
FVAR[14],[15],[16]=cc.Tx,Ty,Tz, FVAR[17],[18],[19]=cc.Rx,Ry,Rz
parm = 0: get camera location in world coordinates: IVAR[10]=cc.ipol,
FVAR[10]=cc.f, FVAR[11]=cc.kappa1, FVAR[12],[13]=cc.p1,p2,
FVAR[14],[15],[16]=x,y,z, FVAR[17],[18],[19]=Rx,Ry,Rz (angles)
IVAR[11] = angles valid = 0|1. Achtung: resets position
parm = 1: f = val1, kappa1 = val2
parm = 2: val1 = p1, val2 = p2: calibration constants
parm = 3: val1 wird als Interpolationsart verstanden: 0|1|2 = keine, linear, spline
parm = 4: set coordinates: Tx = val1, Rx = val2
parm = 5: set coordinates: Ty = val1, Ry = val2
parm = 6: set coordinates: Tz = val1, Rz = val2
type = 3: Set calibration data:
parm = -1: val1 = 0|1 = coplanar|non-coplanar
val2 = -1 | 0 | 1..MAX_POINTS:
point_count=val2, (-1=no change, 0: set point_count to Rt.count)
Then transfer 3d object points and 2d marker locations to calibration data
parm = 1: val1 = 0|1 = coplanar|non-coplanar, val2= point_count
parm = 2: val1 = i [0..point_count], set Xw[i]=val2
parm = 3: val1 = i [0..point_count], set Yw[i]=val2
parm = 4: val1 = i [0..point_count], set Zw[i]=val2
parm = 5: val1 = i [0..point_count], set Xf[i]=val2
parm = 6: val1 = i [0..point_count], set Yf[i]=val2
type = 4: Transform Result Points:
parm = 1: camera coordinates -> world coordinates; n = point count
parm = 2: world coordinates -> camera coordinates; n = point count

Histogrammausgleich (Manual Kapitel 7.2.):

equalize	1, B	use_old: integer; Histogramm-Equalization: Eine Kontrastverstärkung durch gleichmäßige Ausnutzung des Grauwertebereiches, d. h. wenn große Bereiche des Bildes geringen Kontrast mit wenigen Grauwertunterschieden aufweisen, so werden die Grauwerte in diesem Grauwertbereich gespreizt. Wenn use_old zu 1 gesetzt wird, so wird hierzu die beim letzten Mal berechnete Look-Up-Table benutzt. Dies ist sinnvoll, wenn über einen bestimmten Bildausschnitt ein optimaler Equalize-Effekt gefunden wurde und dieser auf das gesamte Bild angewendet werden soll, bei dem sich ja bei normaler Anwendung ein anderes Ergebnis einstellen würde.
scale_hist	0, B	kein Parameter; Skalierung der Grauwerte eines Bildausschnittes auf die vorhandene Grauwertanzahl [0..255] bei Bildern, die den Grauwertebereich nicht ausnutzen, bewirkt dies eine Kontrastverstärkung. Die Funktion arbeitet nach der Formel: Active = (Active - sub_const) * mult_const derart, daß der insgesamte Grauwertebereich genutzt wird. In IVAR[10] wird der Wert der benutzten sub_const übergeben, in FVAR[10] der Wert der benutzten mult_const. Return_Value immer 1.
equalize_ahe	1, B	n: integer; Adaptive Histogrammequalisation. n ist die Anzahl von Feldern in Row/Columns

Arithmetische Funktionen (Manual Kapitel 7.3.):

const_op 8, B

type, iconst: integer; fconst, dummy: floating point; Bild-Operation mit einer Konstanten
type = 0: clear image (clear_roi ist schneller, siehe Löschfunktionen)
type = 1: invert Image
type = 2: add iconst
type = 3: subtract iconst
type = 4: multiply fconst
type = 5: subtract iconst first, then multiply with fconst
type = 6: Image = MaxGray * power(Image/MaxGray, fconst)
type = 7: arithmetisches AND mit der Bit-Konstante iconst.
type = 8: arithmetisches OR mit der Bit-Konstante iconst.
type = 9: arithmetisches XOR mit der Bit-Konstante iconst.
type = 10: shift left << um iconst.
type = 11: shift right >> um iconst.
type = 12: NAND mit der Bit-Konstante iconst.: Src = ~Src & iconst
type = 13: add cyclic iconst, fconst=not used, dummy = exclude = -1|0: -1:don't exclude.
dummy==-1 if Src>255 Src-=256; if Src<0 Src+=256
dummy==0 if Src>255 Src-=255; if Src<1 Src+=255 (excl.0, and don't add to 0)

src_dst_op 2, B

type, const: integer; Bildoperationen mit zwei Bildern
type = 1: add source und destination, Active image = Active + Second
type = 2: sub destination from source, Active image = Active - Second
type = 3: multiply source und destination, Active image = Active * Second / 128
type = 4: divide source und destination, Active image = (Active / Second) * 128
type = 5: sub dest from source and add const, Active image = Active - Second + const
type = 6: add source und dest and add const, Active image = Active + Second + const
type = 7: Bestimme Leistungsspektrum: Active = sqrt(Active*Active+Second*Second)
type = 8: Subtrahiere zyklische Bilder voneinander, z. B. HUE-Farbverläufe oder
-2PI/+2PI-Sägezahnbilder aus der Interferometrie.
const = 0: für zyklische Sägezahnbilder, d. h. das Ergebnis der Subtraktion liegt
immer im Bereich -pi..pi = [0..255] nach der Vorschrift:
Result = Active - Second; if (Result<0) Result+=256;
const = 1: für zyklische HUE-Bilder nach der Vorschrift:
if (Active==0 || Second==0) Result = 255;
else { Result = ABS(Active - Second);
if (Result>127) Result = 255-Result; }
type = 9: AND, d. h. Active &= Second
type = 10: OR, d. h. Active |= Second
type = 11: XOR, d. h. Active ^= Second
type = 12: MAX, d. h. Active = MAX(Active, Second)
type = 13: MIN, d. h. Active = MIN(Active, Second)
type = 14: Use Second Buffer as Threshold Surface for Binarization:
Act := (Scnd-Act>0)?0:255
type = 15: Act := Act + ((Act - Second) * (const/1000))

cut_min_max 3, B

min_max, val, roi_full: integer; Abschneiden kleiner oder großer Grauwerte.
Abschneiden aller Werte, die kleiner oder größer als val sind (Setzen auf "val") in der ROI oder im ganzen Bild. full=0: in der aktuellen ROI, full=1: auf dem gesamten Bild. Return_Value immer 1.

abs_zero 0, B

kein Parameter; Darstellen des Absolutwertes bei Bildern, deren Nullwert bei dem Pixelwert 128 liegen

Lineare und nichtlineare Filter (Manual Kapitel 7.4.):

laplace	2, B	type, size: integer; Laplace-Filter verschiedener Stärke mit 3x3- oder 5*5-Nachbarschaft: Parameter type = [1..6]. Return_Value 1. size = 3 oder 5 für 3*3 oder 5*5 Maske type = 1: low type = 2: medium type = 3: high type = 4: ultra, Maske: -1 -1 -1 -1 8 -1 -1 -1 -1 type = 5: optimal, Maske mit verbesserter Transferfunktion: -1 -2 -1 -2 12 -2 -1 -2 -1 type = 6: sharpen laplace (org + x * laplace); x=0.2 bei 3*3, x=1.0 bei 5*5 Bei size=5 sind type [1..4] gleich type 5 optimal
sobel_val_dir	3, B	neigh248, mode, threshold: integer; Sobel-Filter zur Gradienten- und Winkelsuche. neighbour248 = 4|8: Es wird die 4er oder 8er Nachbarschaft benutzt neighbour248 = 2 => 2*2-Sobel, sehr rauschanfällig! mode=0 ergibt den Betrag des Sobelfilters, d. h. eine Hervorhebung aller Grauwertkanten. mode=1 berechnet die Orientierung der lokalen Struktur über die x- und y-Gradienten im Bereich 0..180 Grad. mode=2 berechnet die Orientierung im Bereich 0..360 Grad. Bei der Orientierungsbestimmung wird der jeweilige Winkelbereich auf den Grauwertebereich von 0..255 abgebildet. Damit kann die Richtung der Kanten sinnvoll in einer Farbkodierung mit der HUE-Farbtabelle betrachtet werden. Die Funktion ist rauschempfindlich. Besser ist es, die Funktion "orientation" anzuwenden. threshold (0..128) ist die Minimalschwelle, ab der ein Winkel der Gradientenrichtung anerkannt wird. Werte unter dieser Schwelle werden als Rauschen anerkannt und zu 0 gesetzt.
sobel_xy	2, B	neighbourhood48, x_y: integer; Sobel-Filter in x- oder y-Richtung. ReturnValue 1.
average	1, B,	Maske: integer; Mittelwert-Filter mit 3*3, 5*5, 7*7 bis 19*19-Maske. Parameter ‘‘Maske’’ ist 3,5,7 bis 19. Return_Value immer 1.
gauss	1, B,	Maske: integer; Gauss-Filter mit 3*3, 5*5, 7*7, 9*9 bis 15*15-Maske. Parameter ‘‘Maske’’ ist 3,5,7,9 bis 15. ReturnValue immer 1.
set_convol_3x3	3, B / ID	interactive, n, val: integer; Eingabe der Matrix-Werte für die user_convolution bei "interactive"=0 n: 1 2 3 4 5 6 Koeffizienten der Konvolutionsmatrix 7 8 9 10 = shift : Additive value 11 = divisor : divide result by divisor value Bei "interactive"=1 interactive Wahl der Koeffizienten für eine 3*3-Konvolution mit Dialogbox. Return_Value 1 bei Ausführung der Konvolution oder richtigem Setzen eines der Koeffizienten, 0 bei Abbruch durch den Benutzer, -1 bei falscher Koeffizientenbezeichnung "n".
convolution_3x3	0, B,	kein Parameter; Ausführen einer beliebigen 3*3 Konvolution, deren Werte zuvor mit der Funktion set_convol_3x3 eingegeben wurden. Return_Value immer 1.
convolution_nxm	4, B	type, shift::div, x, y: integer; Allgemeine Konvolution mit bestimmten vorbesetzten Masken (Structuring Element). z. Zt. muß x==y sein (5,7,9,11,13). Unter type=1 sind verschiedene Laplace (Ultra) Filterkernel implementiert, jeweils mit leicht (um 1 Pixel) abgerundeten Ecken. shift::div ist als (shift<<16)|div definiert. Ergebnispixel werden zuerst durch div dividiert, dann wird shift addiert, d. h. es sollte shift=0 und div=1 sein.
relaxation	2, B (I),	Maske, Iterationen: integer; Relaxations-Filter mit der Maske 3*3, 5*5, 7*7 oder 9*9. Parameter ‘‘Maske’’ ist 3,5,7 oder 9. ReturnValue 1 bei erfolgreicher Ausführung, -1 bei USER-Abbruch mit ESC.
mosaik	1, B,	Mosaik-Kantenlänge: integer; Mosaik-Filter mit beliebiger Kantenlänge der Kacheln. Funktion nicht implementiert, Return_Value -1.

median	2, B,	Filterlänge: integer, Richtung: integer; 1D oder 2D Median-Filter der Länge 3, 5, 7, 9. Richtung=1: Vertikal (Filterlänge = [3,5,7,9] Richtung=2: Horizontal (Filterlänge = [3,5,7,9] Richtung=3: 2-dimensional: Kreuz 3*3 bei Länge 5 oder Fläche 3*3 bei Filterlänge = 9. ReturnValue 1 bei erfolgreicher Ausführung, -1, wenn die ROI zu klein ist. Für 5*5-Median bitte rank_filter verwenden.
rank_filter	4, B,	(beliebige Rank-Filter mit 3*3 oder 5*5 Kernel, auch 5*5 Median: siehe Morphologie)
recursive	7, B,	Richtung: integer, Wichtung: double; Rekursiver Filter mit einer Wichtung von 0.0 bis 1.0. Richtung: 1: Horizontal, 2: Vertikal, 3: Horizontal und Vertikal. Return_Value 1.
iir_2nd_order	4, B	iirtype, passtype, quality1000, freq1000: integer; IIR-Filter 2.Ordnung, horizontal von links nach rechts wirkend, nicht phasenlinear iirtype = 1: critical damping iirtype = 2: bessel iirtype = 3: butterworth iirtype = 4: tschebyscheff 0.5dB iirtype = 5: tschebyscheff 1.0dB iirtype = 6: tschebyscheff 2.0dB iirtype = 7: tschebyscheff 3.0dB passtype = 1|2|3|4|5: low pass | high pass | band pass | band cut | all pass quality1000 = integer(Bandquality * 1000.0); Werte > 1000: schmales Band, Werte < 1000: breites Band freq1000 = integer(Grenzfrequenz * 1000.0) Bei Band Pass und Band Cut ist dies die Bandmittenfrequenz. Die Frequenz muß dabei als Wellenlänge angegeben werden, d. h. bei einer Pixelfrequenz von 14.375MHz und zu filternder Frequenz von 4.43MHz erhält man eine Wellenlänge von 3.245 Pixeln. => freq1000 = 3245
color_filter	1, B	smooth_sharp: integer; 4.43MHz-Farbträger-Filter als Tschebyscheff 0 mit .5dB Restwelligkeit smooth_sharp = 0: Quality=2.215 => 2MHz Bandbreite smooth_sharp = 1: Quality=4.430 => 1MHz Bandbreite smooth_sharp = 2: Quality=8.860 => 0.5MHz Bandbreite Bei Einstellung von 0.5MHz Bandbreite sollte der Farbträger gerade noch sichtbar sein, bei 2MHz kann das Bild schon ein wenig an Schärfe verlieren.
even_odd	1, B,	even_odd_both: integer; Errechnen des ersten oder zweiten Halbbildes bei Videobildern im Interlaced-Modus. even_odd_both=0: Even-Halbbild. even_odd_both=1: Odd-Halbbild. even_odd_both=2: Übereinanderdarstellung von Even- und Odd-Bild Die Even/Odd-Darstellung eines Sequenzbildes, das in den Bildspeicher kopiert wurde, funktioniert nur dann richtig, wenn die obere ROI-y-Koordinate (Y1) gerade ist. (Korrigiert? Bitte prüfen!) Return_Value immer 1.
flat_field	4, B	set_0_1, dealloc_exec_init_black_dark, dark, bright: integer; Flat Field Background Correction, using either a black background image (camera closed) or a dark background image (32<gray_average<64) and a bright image (192<gray_average<255): set_0_1:There are two possible buffers for stereo functionality. Normally use 0. dealloc_exec_init_black_dark: -1: dealloc selected buffers; 0 execute flatfield correction; 1: init black image correction; 2: init dark image correction; dark: Image Buffer 1..n: Use Image Buffer for dark/black image bright: Image Buffer 1..n: Use Image Buffer for bright image Correction with dark image: corrected = (org-offs)/gain with: gain = (bright-dark)/(mean of bright - mean of dark) offs = dark-mean of dark * gain Correction with black image: corrected = (org-black)*mean of (bright-black)/(bright-black). Use close lens cap to produce black image. Doesn't work if black image is corrected within camera.
flt_bckgrnd	3, B	dark_bright, size, add_const: integer; Flatten Background mit Angabe der Größe der interessierenden Objekte und ihrer Helligkeit relativ zum Hintergrund. "size" ist der minimale Durchmesser (oder Breite bei linienartigen Strukturen) in Pixelgrößen. Die Objekte werden durch fortlaufende kreisförmige Erosion (bei dark_bright=1) oder

		Dilatation (bei dark_bright=0) entfernt und der so erhaltene objektfreie Hintergrund vom Originalbild subtrahiert. Mehrmaliges Ausführen kann den Effekt verstärken. add_const = -1|0..255: Add constant to result. For -1, add average gray value of image
flt_regress	3, B	type, offset, excl: integer; Entfernen eines überlagerten Graukeiles im Bild. type: 1=horizontal, 2: vertical, 3: both offs: -1: automatic: do not cut negative values, add... 0..255= add offset excl: -1 = none; 0..255: exclude color from calculation
flt_linescan	4, B/I	init, mean_max, roi_fullsize, interactivefile: integer: Background Correction for LineScan Kameras: init = 1: Initialisierung mit gegebenem Grauwertbild (gleichmäßiges Hintergrundbild) init = -1: deallocate internal Buffer init = 0: do correction with actual image (in ROI) init = 2: save correction data init = 3: load correction data mean_max = 0|1: 0: use average of correction image for reference 1: use maximum of correction image for reference roi_fullsize = 0|1: work on ROI or on full image interactivefile = 0|1: use Data/Pathname/Filename or get names interactively Returnwerte: IVAR[10] = start pixel in image, IVAR[11] = line width
marrhil	2, B	lapl, grad: integer; Marr-Hildreth-Operator zur Kantenextraktion nach der Vorschrift: Ergebnis = Sobel AND ZeroCross(Laplace). In der Originalvorschrift werden hierzu der Sobel-Filter mit 4er Nachbarschaft und ein 3*3 Medium- Laplace-Filter verwendet. Hierzu stellt man die Parameter wie folgt ein: lapl = grad = 1. Sinnvoller erscheint die Verwendung des 8er-Nachbarschaft-Sobel- Filters und ein 5*5er Laplace-Filter mit optimiertem Kernel (lapl = grad = 2). Hiermit werden Fehlerkennungen durch Bildrauschen stark vermindert. Es ist durchaus sinnvoll, vor der Bearbeitung noch einen Gauss-Filter mit 3*3 oder 5*5 Kernel anzuwenden. Als Ergebnis erhält man ein Bild mit 1-Pixel breiten Kanten, deren Grauwert proportional zur Stärke des Grauwertgradienten der Kante ist. Dieses Kantenbild kann mit der Synthese-Funktion (in der canny-Funktion enthalten: siehe unten) zu definierten binären Kanten weiterverarbeitet werden.
canny	4, B	type, low_thresh, high_thresh, edge_buffer: integer; Canny-Edge-Operator zur Kantenextraktion mit 3*3 Kernel. Als Ergebnis erhält man ein Bild mit 1-Pixel breiten Kanten, deren Grauwert proportional zur Stärke des Grauwertgradienten der Kante ist. Dieses Kantenbild kann mit der Synthese-Funktion (in der canny- Funktion enthalten) zu definierten binären Kanten weiterverarbeitet werden. Es ist durchaus sinnvoll, vor der Bearbeitung noch einen Gauss-Filter mit 3*3 oder 5*5 Kernel anzuwenden. Die Synthese-Funktion benötigt zwei Grauwertgrenzwerte, low_thresh und high_thresh. Dabei werden zunächst alle Kenten mit einem Grauwert (Gradienten) größer als high_thresh als definitive Kanten eingetragen. Danach werden, ausgehend von diesen gefundenen Kanten, direkte Fortsetzungen dieser Kanten bei kleineren Gradienten hinzugenommen, bis eine Kante mit dem Grauwert unter die low_thresh-Grenze fällt. Hiermit werden schwache Kanten größtenteils eliminiert, wichtige Kanten, die an manchen Stellen durch Bildrauschen zu schwach sind jedoch noch mitberücksichtigt. Die Benutzung der Synthese-Funktion wird mit dem type-Parameter gesteuert: type = 0: Nur Ausführung des Canny-Edge-Operators. type = 1: Ausführung des Canny-Edge-Operators und dann der Edge-Synthese mit den Schwellwerten low_thresh und high_thresh. Das binäre Ergebnisbild liegt im durch edge_buffer angegebenen Bildspeicher. type = -1: Nur Ausführung des Edge-Synthese, falls diese z. B. für eine Auswertung nach Anwendung anderer Operatoren benötigt wird (Marr- Hildreth). Die Edge-Synthese wird mit den Schwellwerten low_thresh und high_thresh durchgeführt. Das binäre Ergebnisbild liegt im durch edge_buffer angegebenen Bildspeicher.
zerocross	1, B	frame: integer; Bestimmung der Nulldurchgänge im Bild (Zero Crossings). Hierzu muß mit einem Laplace-Filter vorher ein Gradientenbild erzeugt werden, dessen Nullwert auf dem intern definierten Grau-Nullwert für Vorzeichenbehaftete Bilder (128) liegt. Als Ergebnis erhält man ein Bild mit 1-Pixel breiten binärisierten Kanten. Wird der Parameter frame = 1 gesetzt, so wird ein 1-Pixelbreiter Rahmen um das Bild gezeichnet.
edge_trace	4, B/I	threshold, overlay, xstart, ystart: integer; In der Regel interaktives Tracen von Kanten im

Bild. Hierzu wird ein Fadenkreuz auf xstart/ystart im Bild gesetzt. Dann startet der Algorithmus und findet die Kante, solange die minimale Grauwertschwelle zu den Nachbarpunkten erfüllt ist.
threshold = 1..128
overlay = 0|1: Bei overlay=1 wird die gefundene Kante in das Originalgrauwertbild eingezeichnet, bei overlay=0 wird nur die gefundene Kante gezeichnet.

Echtfarbbearbeitung (Manual Kapitel 7.5.):

select_activergb	2, B/I	inter, rgb_buffer: integer; Auswahl und Aktivierung eines bereits definierten TrueColor-Bildspeichers. interactive = 0|1
set_RGB_active	2, B (I)	interactive, buffer: integer; Auswählen des RGB-Bildspeichers, der von anderen Funktionen zum Kopieren oder Transferieren angesprochen wird. Das kann z. B. das einzelne Transferieren der Rot/Grün/Blau-Farbebenen in normale Bildspeicher zur Weiterverarbeitung sein. ReturnValue 1 bei definiertem TrueColor-Buffer, sonst -1.
set_RGB_valid	1, B	valid: integer; Validieren des RGB-Bildspeichers, wenn dieser definiert wurde, aber noch kein RGB-Bild geladen wurde. Diese Funktion ist nur notwendig, wenn ein RGB-Bild sozusagen 'künstlich' erzeugt wurde und dann als RGB-Farbbild akzeptiert werden soll. Das kann z. B. das getrennte Laden einzelner RGB-Anteile eines Farbbildes sein. ReturnValue 1 bei definiertem TrueColor-Buffer, sonst -1.
set_RGB_noise	1, B	noise: integer; Setzen eines Noise-Faktors (in Pixelwerten) für die Umrechnung aus dem RGB-Farbraum in den HSI-Raum. Farbwerte, die unter das Rauschen fallen, d. h. deren Unterschiede in RGB-Werten kleiner als der Rauschwert sind, werden als farblos definiert. Sinnvolle Werte liegen zwischen 10 und 20.
conv_rgb_dth332	1, B (IM),	dither_332: integer; Konvertierung eines 24-Bit-RGB-Bildes in ein 8 Bit-Bild in der Dithertechnik oder mit 332:RGB. Das Bild wird nur auf dem Bildschirm aufgebaut und es ist kein Bildbuffer gültig. Danach muß wieder ein gültiger Bildbuffer aktiviert werden (z. B. mit set_active (n)). Wenn das 8-Bit-Ditherbild weiterverarbeitet werden soll, muß es mit der Funktion ‘‘screen_to_buffer (n)’’ in einen gültigen Bildbuffer übertragen werden. Return_Value 1 bei erfolgreicher Umwandlung, -1 bei nicht definiertem Color-Buffer oder nicht vorhandener Dithertabelle. Bei "dither_332" = 1 Konvertierung eines 24-Bit-RGB-Bildes in ein 8-Bit-Bild mit 3 Bit Grün, 3 Bit Rot und 2 Bit Blau-Anteil.
convert_bayer	4, B	gray_rgb, xoffs, yoffs, rgb_buffer: integer; Konvertierung eines Bayer-Pattern-Bildes in ein RGB-Farbbild oder in ein Graustufenbild. gray_rgb = 0: Ergebnis ist ein Grauwertbild gray_rgb = 1: Ergebnis ist ein RGB-Farbbild, es wird bilinear interpoliert gray_rgb = 2: Es wird eine "Smooth Hue Transition" benutzt xoffs und yoffs ist der Offset des Grün,Rot,Blau,Grün-Pixel-Quadrupels von der linken oberen Bildecke, normalerweise immer 0, kann jedoch auch verschoben sein. rgb_buffer ist der Bildspeicher für das Ergebnis, entweder ein Grauwertspeicher oder ein Echtfarbspeicher
conv_palette	1, B	rgb_buffer: integer; Konvertieren eines Palettenbildes in einem normalen Bildspeicher in ein Echtfarbbild in einem Echtfarbbildspeicher. Die Echtfarbspeichernummer wird mit rgb_buffer angegeben.
color_space	2, B	from_to_RGB, type: integer; Umwandlung in unterschiedliche Farbräume from_to_RGB = 0: aus RGB in anderen Farbraum umwandeln from_to_RGB = 1: in den RGB-Farbraum zurückwandeln type = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 = RGB | HLS | HSI | YIQ | UVW | XYZ | U*V*W* | LAB | YCBCR | CMY Durch die Umwandlung wird der entsprechende color_space automatisch gesetzt.
set_color_space	1, B	color_space: integer; Setzen des aktuellen Farbraumes, nicht notwendig nach Aufruf der color_space()-Funktion. color_space = 0: RGB color_space = 1: HLS color_space = 2: HSI color_space = 3: YIQ (YUV) color_space = 4: UVW color_space = 5: XYZ color_space = 6: U*V*W* color_space = 7: LAB color_space = 8: YCBCR color_space = 9: CMY
conv_rgb_hls	0, B (IM),	kein Parameter; Konvertierung eines 24 Bit-RGB-Bildes in ein 24-Bit-HLS-Bild.

		Return_Value 1 bei Umwandlung, -1 bei nicht definiertem Color-Buffer.
conv_hls_rgb	0, B (IM),	kein Parameter; Konvertierung eines 24-Bit-HLS-Bildes in ein 24-Bit-RGB-Bild. Return_Value 1 bei Umwandlung, -1 bei nicht definiertem Color-Buffer.
conv_rgb_hsv	0, B (IM),	kein Parameter; Konvertierung eines 24-Bit-RGB-Bildes in ein 24-Bit-HSI-Bild. Return_Value 1 bei Umwandlung, -1 bei nicht definiertem Color-Buffer.
conv_hsv_rgb	0, B (IM),	kein Parameter; Konvertierung eines 24-Bit-HSI-Bildes in ein 24-Bit-RGB-Bild. Return_Value 1 bei Umwandlung, -1 bei nicht definiertem Color-Buffer.
rotate_hue	1, B,	Rotation: integer; Rotation in [0..360] Grad, wird intern auf den Pixelwertbereich von 0 bis 255 umgerechnet. Der Nullwert bleibt dabei erhalten, da dieser Wert für Pixel ohne Farbe reserviert ist.
show_col_plane	1, B (IM),	Farbauszug: integer-Äquivalent eines Characters; Anzeige und Aktivierung einer der Farbebenen im Farbbildspeicher. Beispiel: show_farbauszug (82) zeigt den ROT-Anteil, da 82 = hex(52) = ‘‘R’’. Folgende Character sind definiert: R, G, B, H, S, I, L, V, siehe auch picCOLOR-Hot-Key's. Der Farbauszug kann allerdings auch mit der Speichernummer angegeben werden: Farbauszug = 0 | 1 | 2 | 3 | 4 | 5 | 6 = Dither-Bild, R, G, B, Secondx, Secondy, Secondz Return_Value 1 bei Anzeige, -1 bei nicht definiertem Color-Buffer.
col_to_roi	1, B/I	plane: integer; copy color buffer plane to regular 2D-image buffer within ROI plane = 0..6: 0=dither, 1=red, 2=green, 3=blue, 4,5,6=2nd color space
roi_to_col	1, B/I	plane: integer; copy regular 2D-image buffer to color buffer plane within ROI plane = 0..6: 0=dither, 1=red, 2=green, 3=blue, 4,5,6=2nd color space
load_dithtab	0, B (IM),	kein Parameter; Laden der Dithertabelle von der Festplatte. Die Dithertabelle wird benötigt, um Echtfarbbilder auf beliebigen Bildschirmfarbauflösungen darzustellen. Hierzu wird das Echtfarbbild in die minimale 8-Bit-Farbauflösung umgerechnet und mit geeigneter Farbtabelle (LUT) dargestellt. Diese Funktion wird automatisch ausgeführt, wenn die Dithertabelle zum ersten Mal benutzt wird. Return_Value 1 nach erfolgreichem Laden, sonst -1.
free_dithtab	0, B	kein Parameter; Freigeben der Dither-Tabelle, die 48kByte lang ist.

Frequenzraumbearbeitung (Manual Kapitel 7.6.):

dft2_ift2	4, B(IM)	Ind, Shift_DC, Hamming_Blackman, fromto_img_fft: integer; 2D-Fouriertransformation und Rücktransformation für 2^n*2^m-Rechtecke. Ind = 1: Hintransformation; Ind = 2: Rücktransformation; Ind = 3: Nur ein Transfer des Bildes in den FFT-Speicher (Realteil) (from_to not used); Shift_DC=0: Transform DC-Coefficient in upper left corner of freq. domain (ROI) Shift_DC=1: Transform DC-Coefficient in the center of freq. domain (ROI) Hamming_Blackman=1: vor der Transformation Hamming Window angewenden. Nach der Rücktransformation wird dieses wieder zurückgenommen. Bei 2 wird statt dessen der Blackman-Filter benutzt. fromto_img_fft: Angabe, ob Ursprungsdaten aus dem Bild oder aus dem FFT-Buffer genommen werden und ob Ergebnisdaten in das Bild oder inden FFT-Buffer gespeichert werden sollen. fromto_img_fft = 0: FFT: hole Bild aus Image-Buffer und speichere transformiertes Bild im FFT-Buffer. IFFT: hole Daten aus FFT-Buffer und speichere rücktransformierte Daten im Image-Buffer. fromto_img_fft = 1: Alle Daten nur aus FFT-Buffer holen und dorthin wieder zurückspeichern. Die FFT-Koeffizienten werden auf den maximalen Pixelwert normiert, indem sie durch 256 dividiert werden. Return_Value 1 nach Umwandlung, 0 nach User-Abbruch bei sinnvollem FFT-Bufferinhalt, -1 bei User-Abbruch während der Umrechnung (ESC) oder bei Speichermangel.
dft2_ift2_mix	4, B(IM)	Ind, Shift_DC, Hamming_Blackman, fromto_img_fft: integer; Mixed-Radix-FFT für beliebige Fenstergrößen. Alle Parameter wie oben.
fft_arith	8,B	type, buf: integer, valr, vali: float; Arithmetische Funktionen mit FFT-Buffern und Konstanten buf: Nummer der FFT-ROI (FFT-Buffer): 0..2 (entspricht den ROIs 0..2) valr, vali: Real- und Imaginaer-Teil der Konstanten für die Add- und Mul-Funktion. type: 0: Clear, type: 1: Add, type: 2: Mul, type: 3: Conj.compl., type: 4: Invers type: 5: Real/Imag->Power/Phase, type: 6: Power/Phase->Real/Im type: 7: Erstelle Gabor-Filter-Satz im FFT-Buffer. Cosinus-Filter (even) wird im Realteil des Speichers abgelegt, Sinus-Filter (odd) im Imaginärteil. Der Winkel des Gabor-Filters kann z. Zt. nur interaktiv eingegeben werden (Gabor-Orientation [-180..180]). Zur Anwendung des Gabor-Filters wird das zu filternde Bild in den Frequenzraum transformiert. Dann wird in einem zweiten FFT-Speicher der Gabor-Filtersatz erstellt und schließlich wird das transformierte Bild mit dem geraden oder ungeraden Filteranteil multipliziert und in den Ortsraum zurücktransformiert. Achtung: für Bandbreiten > 1.0 ist die Verstärkung des DC-Anteils nicht gleich Null! Wenn mit geradem und ungeradem Gabor-Filter gearbeitet wurde und beide gefilterte Anteile in Bildspeichern vorliegen, kann mit der Funktion src_dst_op(7,0) das Leistungsspektrum des Ausgangsbildes berechnet werden. (Eine Grau-Dilatation des Ergebnisses kann sinnvoll sein). valr=centerfreq, vali=bandwidth type: 8: Gauss-Bandpass-Filterung des FFT-Buffers, valr=centerfreq, vali=bandwidth type: 9: Difference of Gaussian: valr=rc, vali=rs are centre, surround radii at e^(-0.5) (pixels); kc, ks are centre and surround gain constants type:10: Band Pass Filter: (valr=centerfreq, vali=bandwidth) type:11: Normalize FFT-Buffer contents (getrennt für Real- und Imaginärteil) Für Gabor-Filter, Gauss-Filter, DOG, und Band Pass heißen die float-Parameter freq und bandwidth (statt valr und vali): 0 < freq < 180 ist die Mittenfrequenz in degree/pixel, d. h. die höchste Frequenz ist eine Schwingung über zwei Pixelbreiten bei freq=180. bandwidth ist die Breite des Filters in Oktaven
fft_src_dst	4,B	type, buf1, buf2, prm: integer; Arithmetische Funktionen zwischen zwei FFT-Buffern: type: 1: Add, type: 2: Sub, type: 3: Mul,

		type: 4: Div: buf1 /= buf2, with minimum Magnitude = prm/1000000.0 (Inverse Filter) type: 5: CrossCorrelation, type: 6: AutoCorrelation type: 7: copy buf1 -> buf2. Beide FFT-Buffer müssen gleich groß sein! type: 8: buf1 = buf1 + buf2.re type: 9: buf1 = buf1 + buf2.im type:10: buf1 = buf1 * buf2.re type:11: buf1 = buf1 * buf2.im type:12: Wiener Filter: buf1 /= buf2, minimum magnitude = prm/1000000.0, Set Wiener Gamma (1.0=Standard, !=1.0 is parameteric Wiener) Set SN average power spectrum of noise = 1000.0 type 4 (Inverse Filter) und type 12 (Wiener Filter) sind nun in SSE2 und AVX implementiert. buf1, buf2: Nummer der FFT-ROI (FFT-Buffer): 0..2 (entspricht ROI 0..2 = x,y,z)
fft_spectrum	3,B	type, factor, shift: integer; Berechnung verschiedener Spektren: type: 1: Amplitudenspektrum, 2: Phasenspektrum, 3: Realteil, 4: Imaginärteil. factor: -1 = automatic, >0: linearer Faktor (Multiplikativ) shift: 0 : kein Shift, Nullwert = 0. 1 : Shift = 127, Nullwert bei Pixelwert 127 -1: Shift = automatic, falls (factor=-1). Das Spektrum wird dann über den vollen Pixelwertebereich [0..255] gespreizt. Return_Value 1 bei Anzeige, -1 bei undefiniertem FFT-Bufferinhalt oder bei User-Abbruch mit ESC.
fft_passfilt	4, B/I	type, thr1, thr2, xyshift: integer; High-, Low- und Bandpassfilter im Frequenzbereich. type: 1: low pass, 2: high pass, 3: band pass, 4: band cut thr1: Grenzharmonische für low- und high-pass oder innere Grenze für band-pass oder band-cut. thr2: äussere Grenze für band-pass und band-cut. wenn thr1 < 0, dann wird die Funktion interaktiv ausgeführt. xyshift ist im Programm folgendermaßen definiert: union { short yx[2]; long xyshift; } yx[0] = y, yx[1] = x, d. h. 1 bedeutet eine Verschiebung in x-Richtung um 1, 65536 ist eine Verschiebung in y-Richtung (nach unten) um 1. Nur wenn xyshift = 0, bleibt der DC-Anteil bei Bandpass und Hochpass erhalten, bei Verschiebungen wird er auf Null gesetzt. Return_Value 1 bei Ausführung, -1 bei undefiniertem FFT-Speicherinhalt oder bei User-Abbruch mit ESC.
fft_shift_space	0, B	kein Parameter; Verschiebe alle Werte im Frequenzbereich um halbe Periodenlänge, d. h. Mittelpunkt verschiebt sich in die Ecken. Diese Funktion ist in der Regel für Tests gedacht, da sie normalerweise während der Fourier-Transformation durchgeführt wird.
shift_space	0, B	kein Parameter; Verschiebe Ortsbereich um halbe Periodenlänge, d. h. Mittelpunkt verschiebt sich in die Ecken. Funktion kann nur ausgeführt werden, wenn FFT-Buffer definiert wurde. Wie die spezielle hamming_window-Funktion ist diese Funktion in der Regel für Tests gedacht. Über den Parameter 'shift_DC' bei der dft2_ift2-Funktion kann diese Verschiebung besser direkt während der FFT-Umwandlung durchgeführt werden.
hamming_window	1, B	hamming_blackman: integer; Anwendung des Hamming-Filters (1) oder des Blackman-Filters (2) auf die aktuelle ROI (im Ortsbereich). Diese Funktion wird in der Regel nur für Tests benutzt, da die Anwendung des Hamming- oder Blackman-Filters viel besser direkt bei der Transformation mit dft2_ift2 oder dft2_ift2_mix gemacht wird.
spot_filter	4, I	sym, radius, del_add, smooth: integer; Interaktiver Frequenzfilter, mit dem bestimmte Frequenzbereiche und Winkelbereiche "ausradiert" werden können. Return_Value 1 bei Ausführung, -1 bei undefiniertem FFT-Bufferinhalt. sym = 0: use filter at pointed location only sym = 1: use positive symmetry relative to DC sym = -1: use negative symmetry relative to DC radius: maximum radius of filter [0..15] (0 means single point) del_add = 0: delete coeeficients by phase linear multiplication del_add <> 0: add constant to coefficients (del_add = 1.0E6*const) smooth = 0: use rectangular filter window smooth = 1: use (1+cos)-shaped filter window

set_reim_spot	2, B	re_im_scrn, off_on: integer; Setzen der benutzten Buffer (Realteil, Imaginaerteil, Bildschirm) für den spot-filter (re_im_scrn in [0..2]) off_on = 0 : switch off - off_on = 1: switch on
freqhist	0, ID,	kein Parameter; Errechnung eines Frequenzhistogrammes des aktuellen Fourier-Buffers. Return_Value 1 bei sinnvollem FFT-Bufferinhalt, sonst -1.
put_fft	8, B	xhar, yhar: integer; valr, vali: float; Setzen einzelner Koeffizienten im FFT-Speicher. valr/vali: Koeffizienten
get_fft	2, B	xhar, yhar: integer; Abfrage einzelner Koeffizienten des FFT-Speichers. Der Realteil wird in FVAR[10] übergeben, der Imaginärteil in FVAR[11].
free_fftbuf	1, B	roi: integer; Freigeben des FFT-Buffers für die aktuelle FFT-ROI [0..2]. Return-Wert ist "1". Wenn der FFT-Buffer nicht definiert war, wird der Wert "0" zurückgegeben.

Pyramidenerzeugung (Manual Kapitel 7.8.):

pyr_gauss	2, B	laplace, average: integer; Erzeugen der Gauss- und Laplacepyramide. laplace = 0: nur Gausspyramide laplace = 1: Gausspyramide im aktiven Bildspeicher, Laplacepyramide im Second Source Bildspeicher (Active != Second) (Undo of Second not possible) average = 0: keine Mittelung (do not use, Informationsverlust) average = 1: lineares Mitteln (do not use, verschiebt das Bild um einen halben Pixel) average = 3: Frequenzreduktion über 3*3 Gaussfilter average = 5: Frequenzreduktion über 5*5 Gaussfilter
pyr_ROI	1, B	level: integer: Einstellung der ROI auf den gewünschten Pyramiden-Level
pyr_scale	2, B	n, interpol: integer; Zurückskalieren von Teilbildern der Gauss- oder Laplacepyramide. n: Ebene der Pyramide, 0 = Originalbild, 1=erste Verkleinerung... interpol = 0: keine Interpolation interpol = 1: schnelle lineare Interpolation interpol = 2: langsame lineare Interpolation (Testfunktion) interpol = 3: Interpolation über kubische Splines

Texturerkennung (Manual Kapitel 7.7.):

texture_synthese	1,B	noise: integer; Synthese verschiedener Texturmuster zum Test von Texturerkennungsroutinen oder zum Füllen von Flächen mit bestimmten Texturen. Return_Value immer 1.
get_tex_param	4, B	alo,ao,aro,al: integer; Setzen der Parameter für eine Textursynthese. alo = links-oben ao = oben aro = rechts-oben al = links
texture_filter	4, B	type, n, mul, step: integer; Statistischer Texturfilter zur Textur-Segmentierung. type = 1: Max-Min-Funktion type = 2: Standardabweichung type = 3: Varianz (Quadrat der Standardabweichung) n: Größe der Matrix mit n*n (kann beliebig groß sein, nur durch Heap-Speicher begrenzt n sollte ungerade sein, wird intern über n := (n/2)*2+1 ungerade gemacht. mul: Multiplikativer Faktor zur Skalierung des Ergebnisses. Wird intern durch Shift erreicht, daher nur 1,2,4,8,16,32,64 sinnvoll. step: Schrittweite, mit der der Filterkernel sich verschiebt. Return_Value 1 bei Ausführung, -1 bei zu kleiner ROI oder User-Abbruch durch ESC.
orientation	3, B (IM)	buf_angle, buf_satur, buf_inten: integer; Orientierungsanalyse mit der Trägheitstensormethode buf_angle: Nummer des Bildspeichers für den Winkelwert buf_satur: Nummer des Bildspeichers für die Ausgeprägtheit buf_inten: Nummer für einen intern benötigten Bildspeicher Es ist sinnvoll, nach der Transformation den Speicher "buf_angle" in den Hue-Speicher eines Echtfarbbildspeichers zu kopieren und den Speicher "buf_satur" in den Saturation-Speicher. Dann kann das Endergebnis in den RGB-Raum gewandelt werden und dann mit "D" (Dithern) als Farbbild angezeigt werden. Die Farbe entspricht dann dem Winkel, die Intensität der Steilheit der Kanten. Die Funktion arbeitet intern im 16-Bit-Modus.
texture_dct	4, B	type, param, step, kernel: integer; DCT-Analyse (Kosinus-Transformation). type = 1: Pixelwert wird zur Ordnung des letzten Koeffizienten mit einer Amplitude größer als der param-Variable [0..4096] gesetzt. type = 2: Pixelwert proportional zum mehr oder weniger bestimmten Auftreten einer deutlichen Orientierung der Textur. Hierzu werden alle Harmonische im Winkel der angegebenen Orientierung untersucht. Die möglichen Orientierungswinkel werden mit der Variable param = 0, 45, 90 angegeben. type = 3: Größe einer Textur: es werden einzelne Harmonische in alle Richtungen gemittelt und der Pixelwert wird proportional hierzu gesetzt. param ist die Harmonische [0..7] oder [0..15]. step kann zu 1 oder 2 gesetzt werden und ist die Schrittweite der Analyse. Sie sollte nur für eine schnellere Vorschau auf 2 gesetzt werden. kernel ist die Größe der DCT-Fenster und kann 8 oder 16 sein. Wenn die gesuchte Textur nicht innerhalb dieser Fenstergrößen feststellbar ist, so kann mit den Pyramidenfunktionen eine Verkleinerung des Bildes erreicht werden. Die Funktion ist noch nicht vollständig getestet. Diese Funktion ist multithreaded (max 2 Threads)

Binärisierung (Manual Kapitel 7.9.):

Bis auf wenige Ausnahmen liefern alle globalen Binärisierungsfunktionen Threshold 1 und Threshold 2 in den Macrovariablen IVAR[10] und IVAR[11] zurück.

bin_params	4, B	type, read_write, prm1, prm2: integer; Setzen oder Abfragen von Binärisierungsparameters: type = 0: prm1 = execute Binärisierung = 0|1 (für alle globalen Binärisierungen) type = 1: prm1 = left_start, prm2 = right_start (0..255): Start der Maxima-Suche type = 2: prm1 = filter type (1=recursive), prm2 = filter iterations (0..n) type = 3: prm1 = cyclic filter (0|1), prm2 = exception value for filter (-1, 0..255) Bei read_write==0 wird lediglich der gewünschte Wert gelesen und in die Macro-Variablen geschrieben. Hierbei wird der unter prm1 definierte Wert nach IVAR[10] geschrieben, der unter prm2 definierte Wert nach IVAR[11]. Return_Value ist immer 1.
binary	4, B	mode, thr1, thr2, excl: integer; Schwellwert-Binärisierung, dabei gibt es die Möglichkeit, einen Exclude-Wert anzugeben, dessen Grauwerte nicht auf Weiß (255) gesetzt werden. Beispiel: Wenn ein Band von 250..10 binärisiert werden soll, so liegt der Grauwert 0 im Band. Ist dieser Grauwert jedoch für Ausnahmewerte reserviert (z. B. bei der Farbraumbehandlung HUE oder bei der Winkelbestimmung, wo 0 kein Winkel bedeutet), so kann dieser Wert mit excl=0 von der Binärisierung ausgeschlossen werden. Im Beispiel erhalten alle Grauwerte zwischen 250 und 10 bis auf 0 den Wert 255, alle anderen werden zu 0 gesetzt. mode: 1: Multi-Threshold (Fixed 4*4-Mask) 2: Random Mask Threshold 3: Binary Threshold - threshold1 = Threshold 4: Binary Band: threshold1 = lower threshold, threshold2 = upper threshold IVAR[10] = thr1, IVAR[11] = thr2. Diese Schwellwerte werden zurückgegeben, da die Funktion intern von beinahe allen anderen globalen Binärisierungsfunktionen verwendet wird, bei denen die Schwellwerte eventuell automatisch ermittelt werden und die dann von Macro-Programmen eventuell weiterverwendet werden sollen.
binaryband	1, ID,	Threshold_Band: integer; interaktives Binärisieren eines Bildes mir Grenze oder Band. Return_Value 1 bei Ausführung der Binärisierung, 0 bei ‘‘Cancel’’.
binary_RGB	0, I	kein Parameter; Binaerisierung eines RGB-Bildes anhand eines Pixeltrainings. (noch nicht implementiert)
bin_hist	1, B,	Rel_Area: integer; Rel_Area ist die Fläche in Prozent der ROI-Fläche, die über der zu bestimmenden Schwelle liegen soll und damit zu weiß gesetzt werden soll.
bin_entropy	0, B	kein Parameter; Globale Binaerisierung über Maximierung der Entropie des Bildausschnittes
bin_bimodal	3, B	catchrange, startmaxok: integer; Binärisierung über eine Schwellensuche im bimodalen Histogramm eines Bildes. catchrange: 1..128: Bereich, in dem nach weiteren Minima des Histogrammes gesucht wird, nachdem ein lokales Minimum gefunden wurde. Wenn der Bereich zu klein gewählt wird, so wird das absolute Minimum eventuell nicht gefunden. Bei zu großem Bereich wird eventuell über das untere Maximum des bimodalen Histogrammes gesprungen. startmaxok = 0|1: 0: when left/right start is not from 0 or 255, and when it starts on a maximum (>epsilon), then search rising slope first before searching for a maximum!Otherwise a maximum that was supposed to be cut off could be falsely found as a maximum. 1: Allow starting the search on a maximum! Filterart und -iterations über bin_params setzen! Art des Filters zur Glättung des Histogrammvektors; 1=rekursive Filter. Iterations: 0..n: Anzahl der auszuführenden Filterungen, sinnvoll im Bereich 5..50
bin_minamax	4, B	catch, startmaxok, dummy, left_right: integer; Binärisierung mittels Minimum-Suche im Histogramm nach der Vorschrift: finde das Minimum nach Suche des ersten Maximum und setze dahin die Schwelle. catch: Fangbereich zum Finden eines größeren Maximum im Bereich des ersten gefundenen Maximum.

		startmaxok = 0|1: 0: when left/right start is on a maximum, then first search rising gradient! 1: Allow starting the search on a maximum! Filterart und -iterations über bin_params setzen! Art des Filters zur Glättung des Histogrammvektors; 1=rekursive Filter. Iterations: 0..n: Anzahl der auszuführenden Filterungen, sinnvoll im Bereich 5..50 left_right: starte von links (kleine Grauwerte) oder von rechts im Histogramm
bin_percent	8, B	catch, startmaxok: integer percentage, left_right: float; Binärisierung über die Schwellenangabe als Bruchteil des ersten Maximums (von links oder von rechts) catch = 0..128 ist der Fangbereich zum Suchen nach dem absoluten Maximum nach Finden eines lokalen Maximums. startmaxok = 0|1: 0: when left/right start is on a maximum, then first search rising gradient! 1: Allow starting the search on a maximum! Filterart und -iterations über bin_params setzen! Art des Filters zur Glättung des Histogrammvektors; 1=rekursive Filter. Iterations: 0..n: Anzahl der auszuführenden Filterungen, sinnvoll im Bereich 5..50 percentage: 0.0..100.0 Bruchteil des Maximums, der als Schwelle definiert wird left_right = 0.0|1.0, Suche des ersten Maximum von links, d. h. von kleinen Grauwerten aus, bei left_right==0.0, Bei 1.0 wird von rechts aus gesucht.
dyn_entropy	2, B	fieldsize, stepsize: integer; Dynamische lokale Binärisierung mit Entropie-Maximierung in der jeweiligen Nachbarschaft. Die fieldsize sollte dem Bildinhalt angepaßt sein, d. h. bei größeren Strukturen sollte eine größere fielsize gewählt werden. Innerhalb der Nachbarschaftsgröße (fieldsize) sollte über das gesamte Bild (ROI) eine einigermaßen sinnvolle Verteilung von hellen und dunklen Bereichen auftreten. Die stepsize kann dazu benutzt werden, die Funktion zu beschleunigen: das Nachbarschaftsfenster (fieldsize) wird jeweils um die stepsize verschoben, d. h. bei einer stepsize von 2 werden nur für jeden zweiten Pixel in x- und y-Richtung neue Grauwertschwellen berechnet. Bilder mit gröberen Strukturen (größerer fieldsize) können meistens auch eine größere stepsize vertragen.
dyn_bimodal	4, B	fieldsize, catch, stepsize, smooth: integer; Dynamische lokale Binärisierung mit Bimodalitätsprüfung des lokalen Histogrammes. Wen das lokale Histogramm bimodal ist, d. h. zwei Maxima hat, so wird die Grauwertschwelle in die Mitte zwischen beide Maxima gesetzt. Wenn das Histogramm nicht bimodal ist, wird der alte Schwellwert benutzt. Die fieldsize sollte dem Bildinhalt angepaßt sein, d. h. bei größeren Strukturen sollte eine größere fielsize gewählt werden. Innerhalb der Nachbarschaftsgröße (fieldsize) sollte über das gesamte Bild (ROI) eine einigermaßen sinnvolle Verteilung von hellen und dunklen Bereichen auftreten. Die stepsize kann dazu benutzt werden, die Funktion zu beschleunigen: das Nachbarschaftsfenster (fieldsize) wird jeweils um die stepsize verschoben, d. h. bei einer stepsize von 2 werden nur für jeden zweiten Pixel in x- und y-Richtung neue Grauwertschwellen berechnet. Bilder mit gröberen Strukturen (größerer fieldsize) können meistens auch eine größere stepsize vertragen. Da die Histogramme oft sehr verrauscht sein können und dann die Bestimmung der Maximuma sehr problematisch ist, kann mit der Angabe von smooth = 1 ein einfacher rekursiver Filter auf das Histogramm angewandt werden. Mit der Angabe von catch kann ausgeschlossen werden, daß der Algorithmus nur ein Nebenmaximum finded, das Hauptmaximum jedoch übersieht: Bei catch=10 wird z. B. nach Auffinden eines Maximums im Abstand von bis zu 10 Grauwerten überprüft, ob ein größeres Maximum existiert. Wenn ja, so wird dieses neue Maximum angenommen und wiederum im Grauwertabstand "catch" nach einem weiteren, noch größerem Maximum gesucht. Auf diese Weise kann praktisch sichergestellt werden, daß das absolute Maximum gefunden wird. Wenn "catch" zu groß gewählt wird, besteht die Gefahr, daß Versehentlich bis zum zweiten Maximum am anderen Ende des Histogrammes gesprungen wird. Sinnvolle Werte für "catch" sind 10 bis 30, je nach Kontrast des Bildes.
dyn_linipol	3, B	type, regionx, regiony: integer; Lokale dynamische Binärisierung nach Anregung aus dem bekannten KUIM-Bildverarbeitungsprogramm, die eine lineare Interpolation benutzt. type = 0: statistischer Ansatz type = 1: edge-strength-basierter Ansatz type = 2: optimierter Ansatz
bin_method	4, B	type, prm1, prm2, prm3: integer; Globale Binärisierung aus dem XITE-Bildverarbeitungsprogramm. Für eine Parameterbeschreibung siehe Haupthandbuch. type =0: Kittler (kein Parameter)

		type =1: Otsu (kein Parameter) type =2: Abutaleb (kein Parameter). Return: IVAR[11]=Average Threshold type =3: Kapur S W (kein Parameter) type =4: Mardia and Hainsworth (prm1 = use_sem_var, prm2 = convergence_limit)
dyn_method	8, B	type, prm1_regionsize: integer; prm2, parm3: float; Lokale dynamische Bionärisierung aus dem XITE-Programm. Für eine Parameterbeschreibung siehe Haupthandbuch. type =0: Parker, (prm1 = region_size) type = 1: ETM, (prm1 = region_size, prm2=step_size, prm3=distance_limit) type = 2: Yanowitz Br, (prm1 = region_size) type = 3: Niblack, (prm1_regionsize = region_size, prm2 = weight) type =4: NakagawaRosenfeld,(prm1=region_size,prm2=std.-dev.limit,prm3=mean_limit type = 5: Taxt Fl. Jain, (prm1 = region_size, prm2=step_size, prm3=training_size) type = 6: White Rohrer 1, (kein Parameter) type = 7: White Rohrer 2, (prm1 = spot_size_limit, prm2=search_vector_length) type = 8: White Rohrer Improved, (prm1 = region_size)
remove_ghost	2, B	invert, val_step_thresh: integer; Löschen von Geisterbildern, die durch schlechte Binärisierung zustandegekommen sind, z. B. bei Bildern mit starkem Rauschanteil. Dies sind helle oder dunkle Stellen, die sehr geringe Grauwertgradienten aufweisen, trotzdem jedoch als Objekte binärisiert wurden. Zur Ausführung dieser Funktion muß das segmentierte, d. h. binärisierte Bild im "Second" Bildspeicher liegen und im aktuellen Bildspeicher muß das Originalbild liegen.
mark_ghost	2, B	invert, val_step_thresh: integer; Markieren von Geisterbildern, die durch schlechte Binärisierung zustandegekommen sind, z. B. bei Bildern mit starkem Rauschanteil. Dies sind helle oder dunkle Stellen, die sehr geringe Grauwertgradienten aufweisen, trotzdem jedoch als Objekte binärisiert wurden. Zur Ausführung dieser Funktion muß das segmentierte, d. h. binärisierte Bild im "Second" Bildspeicher liegen und im aktuellen Bildspeicher muß das Originalbild liegen.

Morphologie (Manual Kapitel 7.10.):

morph_e_d_o_c	4, B	type, bw_gray, shape, size: integer; Morphologische Operationen: Erosion, Dilatation, Opening, Closing. Objekte sind weiß oder heller als ihre Umgebung. type = 0|1|2|3 = erosion | dilation | opening | closing bw_gray = 0|1 = black/white | gray shape = 0|1|2|3|4|5 = horizontal | vertical | angular | diamond | square | circle size ist die Kernel-Größe in Pixeln, immer ungerade, also 3,5,7,...! Der Winkel für die Angular-Morphology kann z. Zt. nur interaktiv in der Dialogbox gesetzt werden.

Statt dessen gibt es auch noch zwei obsolete Funktionen, die jedoch nicht mehr benutzt werden sollten:

morph_e_d	4, B	type, bw_gray, struct_elem, iter: integer; Erosion/Dilatation Objects are white or have larger Pixelvalues type = 0: Erosion type = 1: Dilatation bw_gray = 0: work on black/white image: 0=0, >0 = 1 bw_gray = 1: work on gray level image struct_elem = 0: 3*1 horizontal neighbourhood struct_elem = 1: 1*3 vertical neighbourhood struct_elem = 5: 4 cross neighbourhood struct_elem = 9: 8 connected neighbourhood iter: 1..n number of iterations
morph_o_c	4, B	type, bw_gray, struct_elem, iter: integer; Opening/Closing type = 0: Opening type = 1: Closing (Alle anderen Parameter wie oben bei morph_e_d)
rank_filter	4, B	type, bw_gray, size, rank: integer; Rank-Filter (siehe auch Kapitel Filter) type = 1: 3*3 cross (-1 <= rank <= 5) type = 2: 3*3 square (-1 <= rank <= 9) type = 3: 5*5 square w/o corner pixel = circle (-1 <= rank <= 21) bw_gray = 0: black/white image bw_gray = 1: gray level image size not used, is controlled by type rank = -1: Dilation rank = 0: Median rank = 1: Erosion rank > 1: Rank-Filter
outline	0, B,	kein Parameter; Alle Schwarz/Weiß-Kanten eines Binärbildes werden auf den Pixelwert 255 (Weiß) gesetzt. Return_Value immer 1.
skeleton	1, B,	method: integer; Es werden iterativ die Skelettlinien aller weißer Flächen eines Binärbildes gesucht. Dabei kann das Ergebnis durch die Wahl der Methode leicht verändert werden. method = 1: normale 4-zyklus Methode method = 2: (nicht implementiert) method = 3: Rosenfeld 4-zyklus-Methode method = 4: Hilditch-Methode (die schnellste Methode), multithreaded (max 2 Threads) method = 5: Verfahren nach Zhang und Suen, kann Linien mit mehr als einer Pixelbreite erzeugen (langsam, nicht optimiert) method = 6: Verfahren nach Lee und Chen, optimiertes Verfahren von Zhang und Suen, alle Linen haben nun Pixelbreite, (langsam, nicht optimiert) Return_Value 1 bei Abschluß der Funktion, -1 bei User-Abbruch mit ESC.
prune_all	0, B	kein Parameter; Abschneiden aller Skelettseitenarme, funktioniert nicht bei 8-connected Ecken, aber sonst ganz gut...
prune	1, B (I)	n: integer; Pruning (=Entfernen) von Skelettseitenarmen der Länge 'n' pixel. Eine Skelettierung muß zuvor durchgeführt worden sein, da nur 1-pixel breite Arme entfernt werden.
remove_single	1, B	type: integer; Entfernen von Einzelpixeln oder kleinen Pixel-Clustern: type = 1: remove single pixel

		type = 2: remove clusters of up to 2*2 pixel Das Bild muß zurvor binärisiert werden. Es werden helle Pixel (Pixelwert 255) entfernt.
remove_small	1, B	size_threshold: integer; Löschen weißer Objekte kleiner als der size_threshold (in Pixel Area)
edm	1, B	euclid_4_8: integer; Distance Map eines Binärbildes. Hierbei kann entweder die Euclidian Distance berechnet werden oder die Distance bei 4er oder 8er-Nachbarschaft (Connectivity). Hierzu ist der Parameter euclid_4_8 zu 1, 4, oder 8 einzustellen. Die resultierenden Pixelwerte stellen die Entfernung vom Rand eines Objektes dar. Die Objekte müssen weiß sein, Pixelwert 255.
watershed	2, B	connect_4_8, boundary: integer; Watershed Funktion zur Trennung von Objekten, die sich berühren oder überlappen. Mit connect_4_8 kann die Art der benutzten Nachbarschaft eingestellt werden (4er oder 8er Connectivity), mit boundary hat man die Wahl, ob die Wasserscheiden bis zum Rand des Bildausschnittes gehen sollen oder ob der komplette Rand des Bildausschnittes auf eine einheitliche Ebene gesetzt wird (boundary=1). Die Vorgehensweise zur Segmentierung ist wie folgt: 1. Man nehme das Originalbild und sichere es auf einem zweiten Bildspeicher. 2. Binärisierung des Bildes, Objekte sind weiß und berühren sich teilweise. 3. Erstellung der Distance Map mit der Funktion edm (hierbei ist Euclidian oder 4er Nachbarschaft sinnvoll) 4. Invertieren des Ergebnisses. Nun hat man ein helles Bild mit den Objekten als etwas dunklere Senken dazwischen. 5. Ausführen der Watershed-Funktion (8er Nachbarschaft, wenn die Objekte später mit der Objekt-Suchfunktion Klassifiziert werden sollen, da diese Funktion auch mit der 8er Nachbarschaft arbeitet) 6. Das Ergebnisbild wird binärisiert mit der festen Schwelle (Threshold) von "1". Man erhält ein weißes Bild mit einem Netz von schwarzen Wasserscheiden. 7. Schließlich wird die logische "AND"-Funktion mit dem binärisierten Originalbild und dem Ergebnisbild ausgeführt. Die schwarzen Wasserscheiden trennen die sich berührenden oder überlappenden Objekte. Die watershed-Funktion benötigt relativ viel Speicherplatz auf dem Heap. ReturnValue ist -1 bei Fehler, z. B. bei knappem Speicher.
labeling	1, B	connect_4_8: integer; Klassifizierung von Objekten in Binärbildern. Dabei werden den Objekten fortlaufende Pixelwerte zugeordnet, beginnend mit 1 oben links im Bildausschnitt. Objekte werden dabei als zusammenhängend nach der 4er oder 8er Nachbarschaft angesehen (connect_4_8 = 4 oder 8).
mark_peak	0, B	kein Parameter; Mark maximum peaks with dot of gray level 255. For plateaus using center of gravity of plateau.
region_grow	4, I	grow_threshold, kernel, xseed, yseed: integer; Region-Growing nach einem interaktiven Seeding mit dem Fadenkreuz. grow_threshold = 0..200, z. B. 50 kernel = ...7, 9, 11, ... xseed, yseed: der Ort, an dem über ein Nachbarschaftshistogramm Grauwertschwellen genommen werden.
extrapol_null	2, B	count, kernel: integer; Extrapolation of Null-Regions, starting at the border of these regions. Function still preliminary... count = -1: forever count = 1..n Iterate n-times kernel = 3 or 5 Schwarze (Null)-Regionen werden vom Rand aus mit den Randpixelwerten aufgefüllt. Der count==-1 bewirkt, daß die Regionen komplett gefüllt werden. Neue Pixel bestimmen sich aus den von Null verschiedenen 3*3 oder 5*5-Umgebungen des jeweiligen Randpixels.

3D/Sequenz-Bildverarbeitung (Manual Kapitel 7.11. und 7.12., Zusatzmodul)

def_3d_image	4, B	w, h, d, acq_buffer: integer; Definiere 3D/Sequenz-Speicher (siehe 3D-Funktionen) w = Breite in Pixeln, h = Höhe in Pixeln, d = Tiefe in Pixeln (3D) oder Sequenzlänge in Bildern acq_buffer = 0|1: Man hat die Wahl, die Möglichkeit zum Gebrauch als Acquisition-Speicher abzuschalten, falls der FrameGrabber diese Möglichkeit bietet. Return Value = 0, falls die allocierte Größe nicht der geforderten entspricht, weil z. B. schon ein 3D/Sequenz-Speicher existiert (alle müssen die gleiche Größe aufweisen). Wenn schon ein Bild definiert wurde, so haben die Parameter keine Wirkung: das neu definierte Bild hat gleiche Größe und Länge.
free_3d_image	0, B	kein Parameter; Abmelden des letzten Sequenz/3D-Bildes und Freigeben des Speichers. Return_Value 1 bei Freigeben, -1, wenn kein 3D/Sequenz-Buffer definiert ist oder wenn andere Buffer zuerst freigegeben werden müssen, auch -1, wenn dieser letzte Speicher gerade von einer FrameGrabber-Karte in Anspruch genommen wird.
get_3d_image	0, B	kein Parameter; Get 3D-image definitions: IVAR[10] = 3d-width IVAR[11] = 3d-height IVAR[12] = 3d-depth IVAR[13] = 3d-pixelsize IVAR[14] = 3d-image-count IVAR[15] = 3d-active IVAR[16] = 3d-destin Wenn kein 3D/Sequenz-Bild definiert ist, dann ist IVAR[14]=0. Return_Value immer 1.
select_active3d	2, B/I	inter, buf: integer; Auswahl und Anzeige eines 3D/Sequenz-Bildspeichers. inter = 1: interaktive Dialogbox inter = 0: nicht interaktive Funktion buf: ist der Sequenz-Bildspeicher [1..n]. Wenn ein nichtexistenter Speicher gewählt wird, wird -1 als ReturnValue zurückgegeben.
set_3d_active	1, B,	3d_buffer: integer; Setzen des intern aktivierten 3d-Buffers, aus dem z. B. durch 2D Bildspeicherfunktionen aus gelesen oder geschrieben wird. (1 <= 3d_active <= 3d-image-count)
set_3d_destin	1, B,	3d-buffer: integer; Setzen des zweiten 3d-Buffers für duale Funktionen (1 <= 3d_destin <= 3d-image-count)
roi_to_d3d	1, B	z_plane: integer; Kopiere ROI-Inhalt in den 3D/Sequenz-Speicher.
d3d_to_roi	2, B	xy_yz_zx, plane: integer; Transferfunktion für Bilder aus dem 3D/Sequenz-Speicher in die normalen Bildspeicher. xy_zy_xz = 0|1|2 ist die Ebene im Raum, die übertragen wird. IVAR[10] ist die Bildnummer, d. h. = plane, IVAR[11] ist der Time-Code/Time-Stamp des Bildes, der bei der letzten Bildübernahme über eine CallBack-Funktion geschrieben wurde, IVAR[12] der Time-Code der zweiten Kamera bei Stereo-Bildeinzug. Wenn der Time-Code als laufende Nummer definiert wurde und nicht der Bildnummer (plane) entspricht, so ist dies ein Zeichen dafür, daß beim Bildeinlesen Bilder verlorengegangen sind.
d3d_show	2, B	plane, show: integer; Aktivieren und Anzeigen einer 2D-Ebene eines Sequenz- oder 3D-Bildes. Dies Funktion dient dazu, ein gewünschtes Sequenzbild zu Aktivieren, um es dann mit den üblichen 2D-Funktionen bearbeiten zu können. Hiermit kann man den Umweg über das Kopieren in einen normalen 2D-Bildspeicher sparen und damit eine Sequenzauswertung wesentlich beschleunigen. Achtung, noch nicht alle 2D-Funktionen sind zur Ausführung mit 2D-Schnitten einer Sequenz oder eines 3D-Bildes geeignet! plane = 0..D3D_depth-1: Aktivieren eines 2D-Schnittes einer Sequenz oder eines 3D-Bildes. Returnwerte: IVAR[10]=LastSeqImage, IVAR[11]=SeqImageTime, IVAR[12]=SeqImageTime2(stereo), IVAR[13]=SeqImageStatus. show = (0|1): Anzeige des aktivierten Bildes.

pick_3dseq	1, I	count: integer; Anzeige jedes (count=1) oder jedes zweiten (count=2) oder bis zu jedem 16ten Bildes der aktiven Sequenz auf dem Bildschirm und Auswahl eines der Bilder mit der linken Maustaste. [1<=count<=16]. Hierbei wird temporär ein neues picCOLOR-Bildfenster geöffnet.
test_3d	2, B	type, const: integer; Erzeuge Testbild im 3D/Sequenzspeicher: type = 0: Running Dot (3*3-Pixel-Block) type = 1: Moving Gray Slope type = 2: Movie-Clip, d. h. Uhr mit Zeiger und Bildnummer const not used
const_3d	8, B	type, iconst: integer; fconst, dummy: float; Konstantenoperationen mit einem 3D/ Sequenzbild. type = 0: clear 3d-image type = 1: invert 3d-image type = 2: add iconst type = 3: subtract iconst type = 4: multiply fconst type = 5: subtract iconst first, then multiply with fconst type = 6: Image = MaxGray * power(Image/MaxGray, fconst) type = 7: arithmetisches AND mit der Bit-Konstante iconst. type = 8: arithmetisches OR mit der Bit-Konstante iconst. type = 9: arithmetisches XOR mit der Bit-Konstante iconst. type = 10: shift left << um iconst. type = 11: shift right >> um iconst. type = 12: NAND mit der Bit-Konstante iconst.: Src = ~Src & iconst not yet implemented: type = 13: add cyclic iconst, fconst=not used, dummy = exclude = -1|0: -1:don't exclude. dummy==-1 if Src>255 Src-=256; if Src<0 Src+=256 dummy==0 if Src>255 Src-=255; if Src<1 Src+=255 (excl.0, and don't add to 0)
src_dst_3d	2, B	type, iconst: Operationen mit zwei 3D/Sequenz-Speichern: type = 0: "Copy: Dst:=Src"; type = 1: "Add: Src+=Dst" type = 2: "Sub: Src-=Dst" type = 3: "Mul: Src*=Dst/128" type = 4: "Div: Src/=Dst*128" type = 5: "Active = Active-Second + const" type = 6: "Active = Active+Second + const" type = 7: "Active = sqrt(A*A+S*S)" type = 8: "Sub cyclic" type = 9: "AND: Src&=Dst" type = 10: "OR: Src|=Dst" type = 11: "XOR: Src^=Dst" type = 12: "MAX: Src=MAX(Src,Dst)" type = 13: "MIN: Src=MIN(Src,Dst)" Return Value immer 1, wenn mindestens ein 3D/Sequenz-Speicher definiert ist.
ppop_3d	4, B	Pixel-Operation über ein 3D-Bild, nicht implementiert, nur für picCOLOR V2.17-Karte
mean_3d	3, B	avr_int, from, to: integer; Mitteln oder Integrieren über mehrere Bilder im Sequenz- oder 3D-Bildspeicher über beliebige Bildanzahlen. Nicht auf 2**n beschränkt. Da die Funktion intern 16Bit-weise arbeitet, muß bei einer Sequenz- oder 3D-Buffer-Einstellung auf 16 Bit dieser vorher gelöscht werden, was automatisch geschieht. Wenn avr_int = 1, dann wird nicht durch die Anzahl der Bilder dividiert (Achtung, overflow möglich!). Das Ergebnisbild wird in die aktuelle ROI auf den aktiven Bildspeicher geschrieben. "from" und "to" sind Start- und Endbilder im Sequenz-Bildspeicher (0<=from<MAX und 1<=to<=MAX).
rotate_3d	1, B	anzahl: integer; Rotiere den Sequenz-Speicher, der hierbei als Ring-Speicher arbeitet. anzahl == -1: Automatisches Rotieren direkt nach einer Sequenz-Bildaufnahme bei Ringbuffer-Einstellung. 0 < anzahl < Sequenzspeicherlänge: Rotieren zum Sequenzspeicherbeginn hin um "anzahl" Bilder.

rank_3d	2, B	rank, size: integer; 3D-Rank-Filter rank = -1: Dilation rank = 0: Median rank = 1: Erosion rank > 1: Rank-Filter size = 3: 3*3*3 cube (dann kann rank: -1 <= rank <= 27 sein) size = 5: 5*5*5 circle/sphere (dann kann rank: -1 <= rank <= 57 sein)
binary_3d	3, B,	type, const1, const2: integer; Binaerisierungs-Operationen für 3D-Bilder: type = 1: threshold at const1 type = 2: binarize band from const1 to const2
convol3_3d	1, B,	type : integer; Konvolution eines 3D-Bildes: type = 4: Laplace type = 6: Sharpen image using Laplace type = 7: Sharpen image very much by Laplace type = 8: Sobel x-direction (not impl.) type = 9: Sobel y-direction (not impl.) type = 10: Sobel z-direction type = 11: Average filter type = 12: Gauss filter
histo_3d	4, B/I	inter, x, y, excl_color: integer: Berechne ein Histogramm eines 3D-Vektors in z-Richtung (oder Zeitrichtung) des Bildes. Rückgabewerte: IVAR[10] = Min IVAR[11] = Max IVAR[12] = Pixelanzahl FVAR[10] = Mean FVAR[11] = Variance FVAR[12] = Entropy
get_voxel	3, B	x,y,z: integer;
put_voxel	4, B	x,y,z,value: integer;
def_3d_fft	1, B	size: integer; definiere kubischen 3D-FFT-Speicher für den aktiven 3D-Buffer. Pro 3D-Bild kann ein einziger 3D-FFT-Speicher definiert werden. Diese 3D-FFT-Speicher werden dann genauso wie die 3D-Bilder mit 3D-Source und 3D-Destination angesprochen. Sie sind immer von der vorderen, linken, oberen Ecke des 3D-Speichers definiert, falls dieser größer ist als der 3D-FFT-Buffer. size = [32, 64, 128, 256, 512]
free_3d_fft	0, B	kein Parameter; Freigeben eines zuvor definierten 3D-FFT-Speichers
dft3_ift3	4, B	Ind, Shift_DC, Hamming_Blackman, From_To_Image: integer; 3D-Fouriertransformation Ind = 1: Forward FFT Ind = 2: Inverse FFT Ind = 3: Transfer Image (Real Part) to FFT Buffer Hamming_Blackman=1: vor der Transformation Hamming Window angewenden. Nach der Rücktransformation wird dieses wieder zurückgenommen. Bei 2 wird statt dessen der Blackman-Filter benutzt. Für alle anderen Parameter siehe die dft2_ift2-Funktion im Kapitel Frequenzraum-bearbeitung.
fft3_pass	3, B,	type, low, high: integer; Pass-Filter für 3d-Bilder type = 1: Low pass using harmoic "low" type = 2: high pass using harmonic "high" type = 3: band pass using "low" and "high"
fft3_spect	1, B,	type: integer; Spectrum-Berechnung für 3d-Bilder type = 1: power spectrum type = 2: phase spectrum type = 3: real part type = 4: imaginary part

fft3_arith	8,B	type, dummy: integer, valr, vali: float; Arithmetische Funktionen mit 3D-FFT-Buffern und Konstanten type: 0: Clear, type: 1: Add, type: 2: Mul, type: 3: Conj.compl., type: 4: Invers type: 5: Real/Imag->Power/Phase, type: 6: Power/Phase->Real/Im type: 7: Erstelle Gabor-Filter-Satz im FFT-Buffer. Der Cosinus-Filter (even) wird im Realteil des Speichers abgelegt, der Sinus-Filter (odd) im Imaginärteil. Der Winkel des Gabor-Filters kann z. Zt. nur interaktiv eingegeben werden (Gabor-Orientation [-180..180]). Zur Anwendung des Gabor-Filters wird das zu filternde Bild in den Frequenzraum transformiert. Dann wird in einem zweiten FFT-Speicher der Gabor-Filtersatz erstellt und schließlich wird das transformierte Bild mit dem geraden oder ungeraden Filteranteil multipliziert und in den Ortsraum zurücktransformiert. Achtung: für Bandbreiten > 1.0 ist die Verstärkung des DC-Anteils nicht gleich Null! Wenn mit geradem und ungeradem Gabor-Filter gearbeitet wurde und beide gefilterte Anteile in Bildspeichern vorliegen, kann mit der Funktion src_dst_op(7,0) das Leistungsspektrum des Ausgangsbildes berechnet werden. (Eine Grau-Dilatation des Ergebnisses kann sinnvoll sein). type: 8: Gauss-Bandpass-Filterung des FFT-Buffers type: 9: Bilde DOG (Difference of Gaussian) des FFT-Buffers type: 11: Normalize FFT-Buffer (Real and Imaginary Part Separately) valr, vali: Real- und Imaginaer-Teil der Konstanten für die Add- und Mul-Funktion. Für Gabor-Filter und Gauss-Filter heißen die float-Parameter freq und bandwidth (statt valr und vali): 0 < freq < 180 ist die Mittenfrequenz des Gaborfilters bzw. Gaussfilters in degree/pixel, d.h. die höchste Frequenz ist eine Schwingung über zwei Pixelbreiten bei freq=180. bandwidth ist die Breite des Filters in Oktaven
fft3_src_dst	4, B	type, src, dst, prm: integer; Arithmetische Funktionen zwischen zwei 3D-FFT-Buffern: type: 1: Add, type: 2: Sub, type: 3: Mul, type: 4: Div: src/= dst, with minimum Magnitude = prm/1000000.0 type: 5: CrossCorrelation, type: 6: AutoCorrelation type: 7: copy src-> dst. Beide FFT-Buffer müssen gleich groß sein! type: 8: src= src+ dst.re type: 9: src= src+ dst.im type:10: src= src* dst.re type:11: src= src* dst.im type:12: Wiener Filter: src/= dst, minimum magnitude = prm/1000000.0, Set Wiener Gamma (1.0=Standard, !=1.0 is parameteric Wiener) Set SN average power spectrum of noise = 1000.0
fft3_shiftspace	0, B	kein Parameter: Verschiebe den DC-Anteil im 3D-FFT-Buffer von der Mitte zur oberen linken vorderen Ecke und zurück.
fft3_polotf	5, B	image_size: float; 3D-polynomial Optical Transfer Function (OTF) image_size ist die Originalbildgröße unter dem Mikroskop in Metern. Die numerische Apertur ist z. Zt. auf NA=1.4 festgelegt.

Geometrische Kontrollfunktionen (Manual Kapitel 8.16.):

meas_control 4, B/I inter, p1, p2, p3: integer; Setzen von Messungsparametern, ersetzt die alten Funktionen set_pol_gauss, set_aver_cross, meas_mono_rgb
inter = -2: Get Parameters:
IVAR[10]=CalUnit,
IVAR[11]=CamCal,
IVAR[12]=PolyWarp
IVAR[13]=RGB_Measure
IVAR[14]=Palette View
IVAR[15]=Polynom_Gauss
IVAR[16]=Average_Cross
IVAR[17]=XCorrelation min size search, default=70 entspricht 0.7x
IVAR[18]=XCorrelation max size search, default=130 entspricht 1.3x
FVAR[10]=Pattern_Matching_Sigma min, default = 2.8
FVAR[11]=Pattern_Matching_Sigma max, default = 2.8
inter = -1: Dialogbox
inter = 0: Set Default Values
inter = 1: Automatische Kalibrierung: p1=cal_unit 0|1, p2=camcal 0|1, p3=polywarp 0|1
inter = 2: p1=RGB-Messung = 0|1, p1=-2: Gray-Calibrated Measure: kalibrierter Grauwert. Hierbei wird in der jeweiligen eingestellten Systemeinheit ausgegeben. Mit der Funktion "color_bar" kann kalibriert werden. Die Systemkalibrierung wird nicht benutzt, sie ist nur für die x- und y-Kalibrierung vorgesehen.
inter = 3: p1 = Palette Display in Status Line 0|1
inter = 4: p1=polynom_gaus 0|1: Setzen des Polynomannäherungs- oder des Gausskurvenannäherungs-Verfahrens für die subpixelgenaue Bestimmung von Peaks im 1D- oder 2D-Frequenzspektrum. 1D wird bei der Linien-Statistik benutzt, 2D bei der Partikel-Image-Velocimetry oder beim Projected Grid Verfahren (Zusatzmodule).
p2=aver_cross 0|1: Art der Subpixel-Maximum-Suche:
aver_cross = 0: es wird der grauwertgewichtete Flächenschwerpunkt der 3*3-Nachbarschafts-Fläche um das absolute Maximum berechnet.
aver_cross = 1: das subpixelgenaue Maximum wird über eine Kurvenanpassung bestimmt. Hierbei wird mit einem Polynom- oder Gaussansatz gearbeitet (siehe oben). Mit den lokalen Maxima über jeweils 3 Zeilen und 3 Spalten in der 3*3-Nachbarschaft um das absolute Maximum werden die Gleichungen zweier Kreise bestimmt, deren Schnittpunkt als Ort des absoluten Maximums genommen wird. Wenn sich nicht zwei Kreise bestimmen lassen, so werden Geradengleichungen benutzt. Diese Art der Subpixelbestimmung ist in der Regel wesentlich genauer.
inter = 5: Set Pattern_Matching_Sigma: p1 = 1000*Sigma.max, p2=1000*Sigma.min
inter = 6: Set XCorrelation min/max size search: p1 = min, p2=max
Value = -1: no change

Data file handling: siehe Input/Output-Funktionen

Geometrische Kalibrierung (Manual Kapitel 8.15.):

set_unit	9, B (IM)	unit_string: String; Eingabe einer Einheit für die Kalibrierung (Üblicherweise vor der Eingabe der Kalibrierung). Wenn noch nicht kalibriert wurde und das System noch auf "pix" eingestellt ist, so kann die Einstellung der Einheit auch nach der Kalibrierung erfolgen, da hierbei keine Umrechnung erfolgt. Systembekannte Einheiten sind: Metrisch: "km, m, dm, cm, mm, um, nm". Andere: "in, ft, mi". Zwischen diesen wird nach Eingabe der Einheit automatisch umgerechnet. Jede beliebige andere Einheit bis zu einer Länge von 3 Zeichen kann eingegeben und benutzt werden, eine Umrechnung und die Abspeicherung mit einem TIF-File ist dann jedoch nicht möglich. Alle intern erkannten Einheiten können mit set_unit(0) abgefragt werden. Es wird dann der String[1]="Recognized Units: ..." zurückgegeben. Die aktuelle Einheit wird in FVAR[10] in Metern zurückgegeben. D. h., wenn die aktuelle Einheit mm ist, wird 0.001 zurückgegeben, wenn die aktuelle Einheit Inch ist, wird 0.0254 zurückgegeben. Zur Abfrage der Einheit ohne Änderung sollte der unit_string leer sein ("") oder man benutze "get_cal_org".
set_cal_org	8, B/I	interactive, cal_org: integer; x, y: float; Setzen der System-Kalibrierfaktoren in [Pixel/Unit] und des System-Ursprunges. Bei "interactive=1" erscheint eine Dialogbox, in der die Kalibrierfaktoren, der Origin, und auch die System-Einheit interaktiv eingegeben werden kann, entweder durch numerische Eingabe oder mit ein oder zwei Fadenkreuzen. Bei "interactive=0" geschieht die Eingabe über die Parameter x,y. cal_org = 1: Eingabe der Systemkalibrierung in [pix/Unit]. Hierbei sollte idealerweise die System-Einheit [Unit] zuvor eingegeben worden sein, da bei nachträglicher Eingabe möglicherweise eine automatisache Einheitenumrechnung vorgenommen wird. Wenn einer der Parameter bei dieser Kalibrierung <=0 ist, wird als ReturnValue -1 zurückgegeben. cal_org = 2: Eingabe des System-Ursprunges (Origin). Hierbei sind nur ganzzahlige Werte sinnvoll, da der Origin intern nur auf Pixelgrenzen gesetzt wird. Negative Werte sind möglich. cal_org = 3: Setzen eines Referenz-Ursprungs in der aktuellen Einheit (für interaktive Eingabe eines Referenzpunktes bei nicht sichtbarem Ursprung). cal_org = 4: Setzen der Rotation des Koordinatensystemes in Radianten [-PI <= x <= PI] cal_org = 5: Einstellung der Y-Koordinatenrichtung: x=1: Y zeigt nach unten; x=-1: Y zeigt nach oben. cal_org = -1: Zurücksetzen der Systemkalibrierung. System-Einheit wird auf [pix] gesetzt, die Kalibrierfaktoren auf 1.0. cal_org = -2: Zurücksetzen des System-Ursprunges auf [0,0]. cal_org = -3: Zurücksetzen des Referenzpunktes auf [0.0,0.0]. cal_org = -4: Rotation des Koordinatensystemes = 0.0. cal_org = -5: Zurücksetzen auf Y-Richtung nach unten. cal_org = 0: Nur Ausgabe der aktuellen Werte in den Macro-Variablen, wie in der Funktion "get_cal_org" beschrieben. ReturnValue 1 bei erfolgreicher Neu-Kalibrierung oder bei sinnvoller alter Kalibrierung nach User-Abbruch, oder 0, wenn nach Benutzer-Abbruch keine Kalibrierung gesetzt ist.
get_cal_org	0, B	kein Parameter; Abfrage der Kalibrierung und der SystemUnit. FVAR[10] = Scale_x in Pixel/Unit FVAR[11] = Scale_y in Pixel/Unit FVAR[12] = Origin_x in Pixel von der linken oberen Ecke FVAR[13] = Origin_y in Pixel von der linken oberen Ecke FVAR[14] = Referenzpunkt (x) in der aktuellen Einheit FVAR[15] = Referenzpunkt (y) in der aktuellen Einheit FVAR[16] = Rotation des Koordinatensystemes in Radianten FVAR[17] = Richtung der Y-Achse (1 unten, -1 nach oben) FVAR[18] = System-Unit in Bruchteilen von [m]: (mm=0.001)
cal_saveload	2, B/I	save_load, interactive: integer; Speichere oder Lade Kalibrierfaktoren, Origin, Rotation, y-Richtung und Unit auf oder von Binaer-File (Extension .CAL). Bei interactive=1 wird der Filename interaktiv eingefragt, sonst wird der gerade aktive offene File benutzt.
color_bar	3, B/I	interactive, horz_vert, smallfont: integer; Kalibrierter Farbbalken für die Zuordnung von kalibrierten Werten zu Grauwerten. Wenn interactive==1, so hängt der Balken am Kursor und wird mit der linken Maustaste plaziert. Ansonsten wird der Ort des einzelnen

		Fadenkreuzes benutzt. horz_vert [0,1] bestimmt die Ausrichtung des Balkens. Mit smallfont=1 kann ein kleiner Font aktiviert werden, sodaß eine größere Anzahl von Nachkommastellen dargestellt wird. In vertikaler Anordnung wird dann zusätzlich Exponentialformat benutzt.
color_bar_cal	8, B/I	autom, physunit: integer; min, max: float; Einstellung der Grenzen und der physikalischen Einheit des Kalibrierbalkens. autom==0: Status der autom-Variable wird nicht geändert. autom > 0: [1|2] Min/Max-Grenzen werden automatisch gesucht. Dies wird von bestimmten Funktionen der Zusatzmodule benutzt. autom==-1: Interpretiere min als Offset, max als Gain-Faktor (noch nicht implementiert) autom==-2: Set Min/Max (Bei der Funktion piv_drawvect werden Werte von min<0 zu min=0 gesetzt) physunit: 0: [-], d. h. keine Einheit, oder benutzerdefinierte Einheit, die nicht der Systemeinheit entsprechen muß, z. Zt. nur interaktiv eingebbar. 1: [Länge=Unit], (Unit ist dabei die jeweilige Systemeeinheit) 2: [Geschwindigkeit=Unit/s], 3: [Beschleunigung=Unit/s^2], 4: [Zirkulation=$Unit^2$/s], -1: [keine Änderung], -2: [interaktive Eingabe] min, max: kleinster/größter Wert des Kalibrierbalkens, jeweils in der aktuellen (oder gleichzeitig gesetzten) physikalischen Einheit. Bei min=max=0.0, werden die Werte nicht geändert. ReturnValue: IVAR[10]=autom, IVAR[11]=physunit, FVAR[10]=min, FVAR[11]=max, FVAR[12]=goffs, FVAR[13]=gfact. Jeder Bildspeicher hat eine eigene Colorbar-Einstellung, die bei Undo jeweils mitzurückgesetzt wird.
scale_bar	8, B/I	interactive, usephysunit: integer; length, mult: float; Kalibrierter Linienzug oder Vektor als Vergleichsgröße zum Anzeigen in einem kalibrierten Bild. interactive = 0: nicht interaktiv, die linke obere Ecke der Box wird an die Stelle des gesetzten Fadenkreuzes gezeichnet. interactive = 1: interaktiv, die Box kann mit der Maus verschoben werden und wird mit der linken Maustaste an der aktuellen Stelle akzeptiert, mit der rechten Maustaste gelöscht. interactive = -1: Box wird in jedem Fall wieder entfernt. usephysunit: die zur Darstellung benutzte physikalische Einheit (siehe color_bar_cal). Bei 'Länge' wird eine begrenzte Linie ausgegeben: |-----|, sonst ein Vektorpfeil: ----->. Bei negativer Eingabe wird der Vektorpfeil erzwungen. length bestimmt die Länge der Linie (des Pfeils) in Pixeln. mult: Multiplikativer Faktor zur zusätzlichen Skalierung der Kalibrierung. Wenn z. B. Vektorpfeile dargestellt sind, die um den Faktor 2 vergrößert dargestellt sind, so muß für eine richtige Kalibrierung hier 1/2 eingegeben werden.

Geometrische Vermessungen (Manual Kapitel 8.1., 8.3.):

single_cross	4, B/I	interactive, rect, x0, y0: integer: Einfache Vermessung mit Fadenkreuz. Setzen eines einfachen Fadenkreuzes oder selektieren eines Rechteckes durch Aufziehen mit dem Mauszeiger. Wenn rect=1, kann mit der linken Maustaste ein Rechteck aufgezogen werden. Nach Loslassen der linken Maustaste Rückgabe der letzten crosshair-Koordinaten in IVAR[10]=x und IVAR[11]=y. Intern im picCOLOR-Programm wird hierbei das einfache Fadenkreuz[0] gesetzt. Nach Aufziehen eines Rechtecks werden die internen Doppelfadenkreuze [1] und [2] gesetzt und deren Koordinaten in IVAR[12]=x1, IVAR[13]=y1, IVAR[14]=x2 und IVAR[15]=y2 zurückgegeben. Wenn kein Rechteck aufgezogen wurde, so sind diese Werte alle = 0. interactive = 1: rect = 0|1: ohne|mit Rechteckfunktion interactive = 0: Setzen des internen einfachen Fadenkreuzes auf x0, y0
cross_hair	4, I	box, savedata, color, buffer: integer; Vermessung mit Fadenkreuz und Zeichnen von Einzelpunkten. buffer: Bildspeicher, in dem gemessen werden soll. Normalerweise wird man im aktiven Bildspeicher messen, es kann jedoch auch Sinn machen, direkt auf dem Bildschirm zu messen, wenn z. B. ein "uneigentliches" Bild dargestellt wird, das in keinem Bildspeicher in dieser Form vorliegt. (-2) <= buffer <= Anzahl der Bildspeicher: buffer == -2 ist der aktuelle Speicher, buffer == -1 ist der Bildschirm, 0<=buffer <= Anzahl ist ein beliebiger Bildspeicher. box=1: Es wird eine Box mit den aktuellen Werten des Fadenkreuzes angezeigt (Koordinaten und Pixelwert Grau oder RGB oder 16 Bit Grau). savedata: Mit diesem Schalter kann bestimmt werden, ob die angewählten Werte auf File abgespeichert werden sollen (natürlich nur, wenn ein Geometry Data-File geöffnet ist). Return_Value = -1, wenn der Schalter "measure_mono_rgb" auf RGB eingeschaltet ist und kein RGB-Bild aktiv ist, sonst immer 1. IVAR[10] = Fx0 (x-Koordinate des Fadenkreuzes) IVAR[11] = Fy0 (y-Koordinate) color=-1: normale Funktion color in [0..255]: bei Punktanwahl mit der linken Maustastewird der Punkt in der Farbe der color gesetzt. Undo ist nach Beendigung der Funktion möglich.
two_cross	4, I	box, connect, stick, savedata: integer; box und savedata wie bei crosshair; connect verbindet die beiden Kreuze mit einer gestrichelten Linie; stick bewirkt, daß sich bei Bewegung eines Kreuzes das andere mitbewegt - stick=1 vertikal, stick=2 horizontal IVAR[10] = Fx1 IVAR[11] = Fy1 IVAR[12] = Fx2 IVAR[13] = Fy2
set_two_cross	4, B	x1,y1,x2,y2: integer; Setzen zweier Fadenkreuze auf eine bestimmte Position (kann auch benutzt werden, um einzelnes Fadenkreuz zu setzen: Werte <0 werden nicht verändert)
ruler	1, I	length: integer; Dreh- und verschiebbarer Maßstab. Länge des Maßstabes in Pixeln [20..600]. Verschieben mit der Maus, Drehen durch Anklicken der linken Maustaste und Verschieben der Maus. Abbruch mit der rechten Maustaste. Wenn das System metrisch kalibriert ist, so wird auch der Maßstab kalibriert dargestellt. Return_Value immer 1.
measline	2, I	Direction, Anzahl: integer; Vermessen einer Länge mit einer Gummibandlinie: Richtung=1: horizontal, Richtung=2: vertikal, Richtung=3: beliebig. Return_Value immer 1. Anzahl gibt die Anzahl der zu vermessenden Linien an. "-1" bedeutet beiebig viele Messungen. Mit ESC oder der rechten Maustaste kann vor Erreichen der durch Anzahl gegebenen Messungen die Funktion abgebrochen werden. In diesem Fall wird der Return_Value "0" zurückgegeben, sonst immer "1". IVAR[10] = Fx1 IVAR[11] = Fy1 IVAR[12] = Fx2 IVAR[13] = Fy2 (der letzen eingegebenen Gerade)
measangle	1, I,	Anzahl: integer; Vermessung von Winkeln mittels zweier Gummibandlinien. Anzahl gibt die Anzahl der zu vermessenden Linien an. "-1" bedeutet beiebig viele Messungen. Mit ESC oder der rechten Maustaste kann vor Erreichen der durch Anzahl gegebenen

		Messungen die Funktion abgebrochen werden. In diesem Fall wird der Return_Value "0" zurückgegeben, sonst immer "1". FVAR[10] = Winkel zwischen den letzen zwei eingegebenen Geraden IVAR[10] = x1a IVAR[14] = x2a IVAR[11] = y1a IVAR[15] = y2a IVAR[12] = x1e IVAR[16] = x2e IVAR[13] = y1e IVAR[17] = y2e
measurecircle	4, I	inter_orig, orgx, orgy, type: integer; Interaktive Vermessung mit zwei Kreisen type = 1|2|3|4 = inner | outer | in & out | in & out inter_orig = 1: Setze Kreismittelpunkt interaktiv mit einem Fadenkreuz inter_orig = 0: Benutze die Werte orgx und orgy als Mittelpunkt
line_edge	4, B/I	interactive, linewide, threshold_mingrad, smooth: integer; Subpixelgeneue Bestimmung von Kanten senkrecht zu einer gewählten Linie. Diese Linie kann eine beliebige Dicke aufweisen. Es werden dann alle Punkte über der Linienbreite gemittelt (pixelgenau senkrecht zur Linie). Bei interaktiver Funktion werden zwei Fadenkreuze zur Wahl der Linie zur Verfügung gestellt. Bei nichtinteraktiver Funktion müssen die Fadenkreuze zuvor mit "set_two_cross" gesetzt werden. 1 < linewide < 32: Linienbreite, über die gemittelt wird. Wenn mit "subpix_prm" thrs_polyorder = 0 gesetzt wurde, so wird threshold_mingrad als Grauwertschwelle interpretiert, sonst als minimaler zu erkennender Grauwertgradient über eine Pixelbreite. Dabei wird die Grauwertkante durch ein Polynoms 3., 5. oder 7. Ordnung angenähert. In diesem Fall muß auch ein sinnvoller Wert für "mingrad" eingegeben werden, da die line_edge-Funktion nicht wie die subpix_area-Funktion nur den maximalen Gradienten sucht, sondern da mehrere Kanten gefunden werden können (bis zu 128). Es werden alle Kanten bestimmt, deren maximaler Gradient größer als der Wert "mingrad" ist. Hierbei hat z. B. eine Grauwertkante mit dem Grauwertunterschied von 50 über den Pixelabstand von Eins den Gradienten 50, falls nicht geglättet wird. Es werden Absolutwerte benutzt. Geglättet werden kann mit Eingabe des smooth-Parameters: 1 führt einmal einen Rekursiv-Filter aus, 2 zweimal usw. Die Funktion gibt die folgenden Werte zurück: IVAR[10] = Anzahl der gefundenen Kanten IVAR[11] = Fadenkreuz 1, x1 IVAR[12] = Fadenkreuz 1, y1 IVAR[13] = Fadenkreuz 2, x2 IVAR[14] = Fadenkreuz 2, y2 FVAR[10] = Echte Länge der Linie in Pixelbreiten FVAR[11] = Skalierte Länge der Linie, falls metrisch kalibriert FVAR[12] = Winkel der Linie zur Horizontalen (skaliert)
edge_data	4, B/I	interactive, n, first_ref, save: integer; Darstellung der gefundenen Kantendaten oder Abfrage der Daten aus dem MACRO-Programm. Bei interactive=0 wird mit "n" angegeben, welche Kante abgefragt wird. (0 <= n < Anzahl_Kanten). Die Daten werden in folgenden Variablen zurückgegeben: IVAR[10] = Vorzeichen des Gradienten, gezählt vom ersten Fadenkreuz aus, d. h. Übergang von hell nach dunkel wird mit "-1" bezeichnet, ein Übergang von dunkel nach hell mit "1". save = 0|1: Abspeichern der Kantendaten auf ASCII-File FVAR[10] = Abstand der Kante vom ersten Fadenkreuz in Pixeln FVAR[11] = Abstand der Kante[n] vom ersten Fadenkreuz in metrischer Einheit, wenn kalibriert wurde. Bei first_ref==1 wird die erste Kante als Referenzkante benutzt und bei der Anzeige oder Ausgabe von allen anderen Kanten subtrahiert. Bei first_ref==0 wird vom ersten Fadenkreuz (durchgezogenes Kreuz) gemessen.

Vektorisierung:

vector	1, I	del_new_data: integer; Interaktive 2-dimensionale Vektorisierung mit der Maus bei del_new_data=1. Der Vektor hängt in Gummibandtechnik am Mauskursor. Mit der linken Maustaste wird ein neuer Punkt bestätigt (hierbei werden die Koordinaten während des Drückens der Maustaste angezeigt). Shift und die linke Maustaste löscht den zuletzt eingegebenen Punkt. Durch wiederholtes Eingeben von Shift und linker Maustaste kann der gesamte Vektorzug gelöscht werden. Beendigung der Funktion mit der rechten Maustaste. Wenn del_new_data = 0, wird der vorhandene Vektorzug gelöscht und der Speicher freigegeben. Bei del_new_data = 2 werden die Vektordaten in einer Dialogbox angezeigt.
vect_plotedit	4, B/I	type,color,fac,edit: integer; Zeichnen und/oder Editieren der Vektordaten. Beim Plotten wird mit der gewünschten Farbe "color" gezeichnet. Bei color=-1 wird invertiert. Beim Editieren (edit==1) wird der Vektorzug invertiert und man kann dann durch Anklicken eines Vektoranfangs oder -endes diesen mit der Maus verschieben. Bildschirm-Zoom-Funktionen (Hotkey <, >) sind während des Editierens aktiv. Bei type=1 werden die Vektorpunkte einfach verbunden, bei type=2 werden Vektorpfeile gezeichnet. Hierbei werden die Punkte 0 und 1 als Vektor aufgefaßt, dann die Punkte 2 und 3, 4 und 5, und so weiter. color kann die Werte [-1..255] annehmen, fac ist ein multiplikativer Faktor zur Änderung der Größendarstellung. fac < 0 bedeutet 1/fac.
vect_saveload	3, B/I	save_load, interactive, start_column: integer; Abspeichern oder Laden eines 3D-Vektorzuges im FloatingPoint-Format. Dabei werden zuerst die Daten des Koordinatenursprunges abgespeichert (x,y,z), dann die Daten des Vektorzuges. Nach 2D-Vektorisierung sind die z-Werte immer 0.0. Z-Werte ungleich 0.0 können jedoch eingelesen werden und dann auch 3-dimensional dargestellt werden. Beim Abspeichern ist "interactive nicht wirksam: Zuerst wird geprüft, ob ein MACRO-File zum Schreiben geöffnet ist, dann wird dieser benutzt. Falls dies nicht der Fall ist, wird geprüft, ob ein Systemdatenfile geöffnet ist. Wenn ja, so wird dieser benutzt. Nur wenn kein Datenfile zum Schreiben geöffnet ist, wird der Filename interaktiv eingefragt. Nur dann wird der File später wieder geschlossen. Beim Einlesen kann angegeben werden, ob ein Filename interaktiv eingefragt werden soll. Dieser wird dann später wieder geschlossen. Wenn interactive=0, so wird ein geöffneter MACRO-File zum Einlesen benutzt. start_column = 0..n: hiermit können Spalten beim Einlesen der Vektordaten übersprungen werden. Achtung: wenn die Gesamtanzahl (übersprungene und zu lesende) der Spalten > 9 ist, muß zuvor mit readfln_maxnum(max) die interne Maximalanzahl der Spalten hochgesetzt werden. Hierbei werden intern die Integervariablen IVAR[10]..IVAR[10+n] überschrieben.
vect_area	0, B(IM)	kein Parameter; Berechnung der eingeschlossenen Fläche eines 2D-Vektorzuges. Hierzu darf der Vektorzug nur schwach konkav sein. Das bedeutet, vom Mittelpunkt des umschreibenden Rechtecks müssen alle Randpunkte mit gerader Linie erreichbar sein. Der Vektorzug wird dabei automatisch mit einer Gerade geschlossen, falls dies nicht schon der Fall ist.
vect_coords	1, B	off_on: integer; Anzeige des Koordinatensystems ein- oder ausschalten.
vect_shiftorig	4, B/I	inter, x, y, z: integer; Verschiebung des Koordinatenursprunges eines Vektordatensatzes beim Einlesen von Festplatte und korrigiere diese Verschieben beim eventuellen späteren Abspeichern. Wenn "inter" = 1 ist, werden die x,y,z-Werte nicht berücksichtigt.
vect_xy_yz_zx	1, B	xy_yz_zx: integer; Einstellung der Vektordarstellung mit der vect_plotedit-Funktion bei 3-dimensionalen Verktordatensätzen. xy_yz_zx kann die Werte 1, 2, oder 3 annehmen und gibt an, in welcher Projektion der Datensatz dargestellt wird. (Funktion nur mit 3D-Zusatzmodul erhältlich)
vect_3d	4, B/I	inter, stereo, vectortype, color: integer; Darstellung von 3D-Vektordatensätzen, eventuell animiert (mit 3D-Zusatzmodul), siehe auch erweiterte Beschreibung im Kapitel Graphik-Funktionen.
vect_transform	7, B	type: integer; value: float; Transformieren eines 3D-Vectors type = -2: Reset all values type = -1: show interactive dialog box type = 0: do rotation type = 1: input translation in x axis

type = 2: input translation in y axis
type = 3: input translation in z axis
type = 4: input rotation about x axis (roll)
type = 5: input rotation about y axis (pitch)
type = 6: input rotation about z axis (yaw)
type = 7: input 2nd rotation about y axis
type = 8: input 2nd rotation about x axis
type = 9: input 2nd translation in x axis
type = 10: input 2nd translation in y axis
type = 11: input 2nd translation in z axis

Flächenbestimmung (Area):

get_pperc	0, I,	kein Parameter; Flächenberechnung eines Pixelwertebandes, das durch den aktuellen Pixel und die definierte Bandbreite gegeben ist. Return_Value immer 1.
set_band	1, B,	Halb_Bandbreite: integer; Setzen der Bandbreite für die Flächenberechnungsfunktion get_pperc. Dabei ist die effektive Bandbreite = 2*(Halb_Bandbreite)+1. Die Halb_Bandbreite darf im Bereich [0..127] gewählt werden. Return_Value 1 bei zulässiger Halb_Bandbreite, sonst -1. Halb_Bandbreite = -1 => get width to IVAR[10] Jeder Bildspeicher hat seine eigenen Werte!
calc_area	3, B / ID	interactive, threshold1, threshold2: integer; Flächenberechnung mit Hilfe eines Pixelwertebandes, das im interaktven Fall mit der Farbtabelle angedeutet wird. Return_Value 1. FVAR[10] = Fläche in Pixeln oder in Meter*Meter, wenn skaliert FVAR[11] = Prozentualer Anteil In IVAR[10] wird die gerundete Pixelfläche zurückgegeben, in IVAR[11] und [12] Start- und End-Threshold.
set_count_null	1, B,	Count_Null: integer; Mitrechnen des Pixelwertes "0" bei der relativen Flächenberechnung, falls dieser in das gewählte Band fällt. Wenn Count_Null = 0, wird der Nullwert mitgerechnet, bei Count_Null = 1 nicht. Return_Value immer 1.
set_band_color	1, B,	Band_Farb: integer; Setzen der LUT in einer bestimmten Farbkombination für die interaktive Flächenberechnung mit LUT. Es gibt 4 verschiedene Möglichkeiten, die mit Band_Farb gewählt werden können: 0: Graukeil mit hervorgehobenen roten Band, 1: HUE-Farbtabelle mit hervorgehobenen weißen Band, 2: dunkle HUE-Farbtabelle mit hervorgehobenen hellem Farbband, 3: First16-Tabelle für Objektklassifikation: die ersten 16 Pixelwerte sind durch stark voneinander abgegrenzte Farben gut unterscheidbar, der Rest wird mit einem normalen Graukeil belegt. Das gewählte Band (üblicherweise bei der Obj.Klassifikation mit der Breite "1") erscheint in der oben angegebenen Farbe, der Rest wird mit schwarz ausgeblendet. Der ReturnValue ist immer 1.

Objektsuche, Blob-Suche:

single_object	4,I	n, man_singl_train, showbox, threshold_box: integer; Vermessung einzelner Objekte oder Partikel (Rejection-Kriterien sind ausgeschaltet). n: Anzahl der Objekte (n=-1: beliebig viele Objekte). man_singl_train = 0: Benutze die System-Objektliste zum Abspeichern der Objektdaten. Die System-Objektliste wird nicht überschrieben, sondern die bestimmten Objekte werden an die Liste angehängt, wenn "append_to_list"-Flag eingeschaltet ist (mit set_append_obj 1). man_singl_train = 1: Benutze Einzel-Variablenstruktur für die Objektdaten, die für jedes Objekt wieder überschrieben werden, d. h. es wird keine Liste erzeugt und die interne Objektliste auch nicht überschrieben. man_singl_train = 2: Trainieren der Rejection

(Klassifizierungs-) Parameter. Die Rejection-Parameter werden durch diese Funktion jedoch noch nicht enabled.
showbox: Anzeige der Datenbox für jedes bestimmte Objekt
threshold_box: [0..255] Grauwertgrenze, ab der ein Objekt erkannt wird. Bei negativen Werten [-255..-1] wird die Grauwertgrenze nach einer Histogrammberechnung als Mittelwert der quadratischen Umgebung der Größe (threshold_box) um den angewählten Pixel angesetzt.
IVAR[10] = Anzahl der angewählten Objekte
IVAR[11] = Grauwertschwelle des letzten angewählten Objektes (für automatische Grauwertschwelle)
ReturnValue = 1 bei richtiger Ausführung, 0 bei ungenügender Anzahl angewählter Objekte. Wenn showbox is on, return value is result of interactive acceptance of last object = -1|0|1.

search_objects 3, B/I

threshold, use_contourbuffer, clearoverlayinroi: integer: Objektsuche: Objekte sind heller oder dunkler als der Wert "threshold", je nach Stellung der Variable Dark_Bright_Objects. Wenn "use_contourbuffer" abgeschaltet ist (0), wird zur Feststellung ob ein Objekt schon gefunden wurde, die oktogonale Umgebung aller schon gefundenen Objekte berücksichtigt. Dies ist wesentlich schneller als die Suche über den Konturenspeicher. Hierdurch kann es jedoch vorkommen, daß Objekte, die vollständig innerhalb der oktogonalen Umgebung eines größeren Objektes liegen, nicht gefunden werden. Vor der Ausführung der Funktion müssen wichtige Parameter mit den (set_xxxx)-Funktionen gesetzt werden. Daten der gefundenen Objekte können mit den Funktionen "object_data" und "ellips_data" abgerufen werden. Die Objektfläche kann nachträglich auch subpixelgenau bestimmt werden. Hierzu dienen die Funktionen "subpix_area" und "subpix_prm".
IVAR[10] = Anzahl der akzeptierten gefundenen Objekte
IVAR[11] = Touch-Border Anzahl
IVAR[12] = Rejected Anzahl
IVAR[13] = Holes Anzahl
IVAR[14] = Not Valid for any reason Anzahl
IVAR[15] = Open Contours Anzahl
IVAR[16] = Already Searched (y < start) Anzahl
IVAR[17] = Unknown (area=0.0???) Anzahl

subpix_area 3, B/I

smooth, threshold, show_area: integer; Subpixelgenaue Bestimmung der Fläche aller zuvor mit der Funktion "search_objects" gefundenen Objekte. Die Ergebnisflächen können mit der Funktion "object_data" abgerufen werden. Mit smooth (0..n) kann der vergrößerte interpolierte Kantenverlauf geglättet werden (rekursiv-Filter). Wenn mit der Funktion "subpix_prm" thrs_polyorder=0 auf threshold eingestellt wurde, so muß mit threshold die Grauwertschwelle eingegeben werden, sonst ist dieser Wert bedeutungslos. Pixel mit dem Grauwert "threshold" gehören dann mit zum Objekt. Ansonsten wird der maximale Grauwertgradient an der Objektkante gesucht.
Bei show_area==1 wird die Pixelfläche zur Überprüfung graphisch dargestellt und muß mit der Maus akzeptiert werden.

object_data 2, B,

n, single_list: integer; Abfrage der gefundenen Objekt-Daten.
0<n<Maximale Anzahl Objekte
single_list = 0: Daten der Single-Variablenstruktur (n ohne Einfluß)
single_list = 1: Daten der normalen internen Objektliste.
1) Werte in Pixelgrößen, Bildspeicherkoordinaten, [0,0]=lefttop:
IVAR[10] = x1 (linke obere Ecke, umschreibendes Rechteck)
IVAR[11] = y1
IVAR[12] = x2 (rechte untere Ecke)
IVAR[13] = y2
IVAR[14] = xm (Center of surrounding rectangle)
IVAR[15] = ym
IVAR[16] = width
IVAR[17] = height
IVAR[18] = boundary length (gerundet, in Pixellängen). Diagonalelemente haben die Länge 1.4142. Achtung: Genauere Boundary in FVAR[19], siehe unten!
IVAR[19] = Fläche in Pixeln
2) Kalibrierte Werte (oder in Pixeln, wenn nicht kalibriert):
FVAR[10] = (x1-Origin_x)/Scale_x (linke obere Ecke)
FVAR[11] = (y1-Origin_y)/Scale_y

		FVAR[12] = (x2-Origin_x)/Scale_x (rechte untere Ecke) FVAR[13] = (y2-Origin_y)/Scale_y FVAR[14] = (xm-Origin_x)/Scale_x (Center of sur. rect.) FVAR[15] = (ym-Origin_y)/Scale_y FVAR[16] = Pixel (oder Subpixel) Area / (Scale_x * Scale_y) FVAR[17] = (xcg-Origin_x)/Scale_x (Center of Gravity) FVAR[18] = (ycg-Origin_y)/Scale_y FVAR[19] = boundary length (according to Smeulders (1989) (see Russ page 518) the optimal length is: length = 0.948*orthogonal + 1.340*diagonal => mean error < 2.5%) Wenn keine Kalibrationsfaktoren gesetzt sind (nicht metrisch kalibriert), so werden Scale_x und Scale_y als 1.0 benutzt.
ellipse_data	2, B	n, single_list: integer; Übergabe der Daten des n-ten Objektes: 0<n<Maximale Anzahl Objekte single_list = 0: Daten der Single-Variablenstruktur (n ohne Einfluß) single_list = 1: Daten der normalen Objektliste. IVAR[10] = xm (Mittelpunkt des umschreibenden Rechtecks) IVAR[11] = ym IVAR[12] = ellipse dmin IVAR[13] = ellipse dmax IVAR[14] = ellipse darea IVAR[15] = ellipse dalpha Außerdem noch die Umrandungslänge in x-,y- und Diagonalenrichtung. IVAR[16] = boundx IVAR[17] = boundy IVAR[18] = boundd IVAR[19] = Wert der internen Variable ‘‘distance’’. Nach einer Objektsuche ist hier die Loch-Information abgespeichert: 0: keine Löcher und 1:mindestens ein Loch. FVAR[10] = ellipse ratio (dmax/dmin) FVAR[11] = Grauwert, mittlerer FVAR[12] = Roundness, using Smeulders boundary length with correction for small and large objects: roundness = (4*pi*area)/(boundary*boundary)*(1/circle_roundness) with circle_roundness = area/(area-sqrt(pi*area)+pi/4) FVAR[13] = Gray-Sigma xm, ym, area, dmin, dmax, boundary werden in Pixelgrößen zurückgegeben
smrect_data	2, B	n, single_list: integer; Übergabe der Daten des kleinsten umschreibenden Rechtecks des n-ten Objektes (smallest rectangle): 0<n<Maximale Anzahl Objekte single_list = 0: Daten der Single-Variablenstruktur (n ohne Einfluß) single_list = 1: Daten der normalen Objektliste. IVAR[10] = Mittelpunkt x IVAR[11] = Mittelpunkt y IVAR[12] = 1. Ecke x IVAR[13] = 1. Ecke y IVAR[14] = 2. Ecke x IVAR[15] = 2. Ecke y IVAR[16] = 3. Ecke x IVAR[17] = 3. Ecke y IVAR[18] = 4. Ecke x IVAR[19] = 4. Ecke y FVAR[10] = Fläche des kleinsten Rechtecks FVAR[11] = Winkel zur x-Achse in Grad Achtung: zur Berechnung der kleinsten umschreibenden Rechtecke müssen die Freeman-Konturen gesetzt sein.
save_objects	2, B/I	single_list, interactive: integer; Abspeichern Objektdaten auf Geometry-Data-File, auf den offenen Systemdatenfile bei interactive=0. single_list = 0: Einzelnes Objekt der "Single Object"-Funktion single_list = 1: Alle Objekte der System-Objektliste
objprm_saveload	2, B/I	saveload, interactive: integer; Abspeichern aller Objektparameter auf einen Datenfile oder Laden von einem File. Vorsicht, diese Parameter findet man in mehreren Dialogboxen verteilt, nämlich der Objekt-Box, der Subpixel-Area-Box, der Objekt-Cut-Box, der RejectionCriteria-Box, usw...
select_feature	2, B	feature, unsel_sel_toggle: integer; Auswahl der abzuspeichernden Objektmerkmale. Auch für Anzeige in der Dialogbox. feature = [0..MaxFeature], Reihenfolge exakt wie im interaktiven Menü, erstes Merkmal ist [0].

unsel_sel_toggle: 0= unselect feature, 1=select feature, -1=toggle feature.
Features:
0 = Xmin, 1=Ymin, 2=Xmax, 3=Ymax, 4=Xcenter, 5=Ycenter
6 = Boundary, 7=RectangleArea, 8=PixelArea,
9=Xcg, 10=Ycg, 11=Roundness, 12=Holes(Löcher),
13=EllipseArea, 14=EllipseDmin, 15=EllipseDmax,
16=EllipseDratio, 17:EllipseAlpha, 18= Gray Mean
19= Gray Sigma, 20 = Boundary x, 21 = Boundary y
22 = Boundary diagonal
23 = MX2, 24= MY2, 25= MXY (Momente)

subpix_prm 4, B

precision, prec_angle, thrs_polyorder, interpol_lin_spl: integer; Subpixel-Parameter:
precision kann zu 1, 2, 4 gewählt werden. Bei precision=1 wird nicht interpoliert, bei precision=2 wird ein Pixel interpoliert, bei precision=4 werden 3 Pixel interpoliert.
prec_angle wird nur für die Objektflächenmessung benutzt: ausgehend vom Mittelpunkt eines Objektes wird die Fläche kleiner Sektoren der Größe (Radius * Delta_angle / 2) bestimmt und aufaddiert. Radius ist hierbei der Abstand vom Objektmittelpunkt zur subpixelgenau bestimmten Kante des Objektes, Delta_angle ist ein sehr kleiner Winkel, für den gilt: sin(Delta_angle)=Delta_angle. Hierbei ist Delta_angle = 1 / prec_angle, wobei prec_angle die Werte 1, 2, 4, 8 annehmen kann. prec_angle sollte immer gleich oder höher als precision sein, d. h. bei höherer Auflösung sollten auch kleinere Winkelinkremente benutzt werden.
thrs_polyorder = 0: Die subpixelgenaue Kante im Bild kann nach der Interpolation der zusätzlichen Funktionswerte über die normale Grauwertschwelle bestimmt werden. Meistens ist es jedoch viel sinnvoller, den steilsten Grauwertgradienten als Objektkante zu definieren. Hierzu setze man
thrs_polyorder = (3|5|7): Es wird der steilste Gradient einer Objektkante gesucht, unabhängig vom Grauwert. In diesem Fall muß mit thrs_polyorder die Anzahl der Stützpunkte des Polynoms eingestellt werden, mit dem der Grauwertverlauf der Kante angenähert und untersucht wird. Es kann 3, 5, oder 7 Stützstellen haben.
Die jeweilige Grauwertschwelle oder bei Linienmessungen auch der minimale zu erkennende Gradient kann bei der jeweiligen Funktion (subpix_area oder line_edge) angegeben werden.
interpol_lin_spl = 0: es wird linear interpoliert
interpol_lin_spl = 1: es wird mit kubischen Splines interpoliert. Achtung: bei sehr starken Grauwertsprüngen (z. B. bei künst-lich erzeugten Bildern) können die Splines überschwingen.
Wenn precision=0 gesetzt wird, so werden keine Werte geändert und die Funktion gibt alle Werte in den folgenden MACRO-Variablen zurück: IVAR[10] = precision
IVAR[11] = prec_angle IVAR[12] = thrs_gradient
IVAR[13] = polynom_order IVAR[14] = interpol_lin_spl

set_pix_area 4, B

area, grayweighted, dark_bright, save_freeman: integer; Setzen von Objektsuch-parametern, Art der Bestimmung von Fläche und "Center of Gravity" (xcg/ycg) bei der Objektsuche:
area = 0: keine Flächenbestimmung und keine Center of Gravity-Bestimmung (Als Ergebnis erhält man die Fläche des umgebenden Achtecks, Xcg = Ycg = 0.0)
area = 1: Bestimmung der exakten Pixelfläche (und Xcg, Ycg und außerdem dem mittlerem Grauwert des Objektes) mit iterativem Ansatz (langsam)
area = 2: Bestimmung der exakten Pixelfläche (und Xcg, Ycg und des mittleren Grauwertes des Objektes) über den ‘‘Pixel-Runs’’-Ansatz. (schnell, kann jedoch zu Fehlern führen, wenn das Objekt mehr als ein Loch oder stark verschlungene oder konkave Löcher enthält.
grayweighted=0|1|2: Zur Bestimmung der Center of Gravity-Werte werden die Pixel mit ihrem Grauwert gewichtet. Bei 2 den maximalen Grauwertpeak nehmen (wenn nicht gefunden, wieder grayweighted nehmen) (area muß hierzu 1 oder 2 sein).
dark_bright = 0|1: 0: suche dunkle Objekte, 1: suche helle Objekte
save_freeman = 0|1; 1: save freeman contour, obligatory for using smallest rect or freeman contour display
Eingabe = -1: no change. Return Values: IVAR[10] = area, IVAR[11] = use pixel runs, IVAR[12] = gray weighted, IVAR[13] = dark_bright

set_append_obj 1, B

clearlist_append: integer; Löschen der Objektliste bei jeder Neubestimmung oder Anhängen neu gefundener Objekte an die vorhandene Objektliste.

set_rejection	8, B	type, clear_set: integer; min, max: float; Setzen der Klassifizierungsparameter "type": 1 = Boundary 2 = Pixel Area 4 = Pixel Area / Box Area 8 = Ellipse Area 16 = Ellipse dmin 32 = Ellipse dmax 64 = Ellipse Ratio dmax/dmin 128 = Ellipse Alpha 256 = Roundness (=area*4*pi/sqr(boundary)) 512 = Gray Mean Value 1024 = Gray Sigma Bei Setzen eines Einzeltypes werden die Werte min und max zugewiesen, bei Kombinationen werden nur die jeweiligen Klassifikatoren enabled, d. h. bei Eingabe von type = 72 wird Ellipse Ratio und Ellipse Area als Klassifikator benutzt, deren min und max Werte jedoch nicht geändert. Bei clear_set=0 wird der jeweilige Klassifikatortype disabled. clear_set = -1: reset all limits for all rejection criteria
set_contour_jump	1, B	contour_jump: integer; contour_jump = 0 bedeutet Suche nach der exakten Kontur, bei Werten > 0 können bis zu "n" Einzelpixel übersprungen werden, um kleine Lücken zu schließen.
set_cutobj	4, B	flag, dist, thr_angle, maxlen: integer; Ein- und Ausschalten der automatischen Objekttrennung und setzen der Parameter, auch für die Konturuntersuchung. - flag: Bit0: 0: Ausschalten, 1: Einschalten der automatischen Trennung, setzen aller Variablen. Bit 1: if set, average two points as a smoothing filter flag<0: Abfragen der Variablen: IVAR[10]=flag, IVAR[11]=dist, IVAR[12]=thr_angle, IVAR[13]=maxlen, IVAR[14]=color - dist: Zur Krümmungsbestimmung der Kontur werden 3 Konturpunkte verwendet. der Abstand dieser Punkte kann eingestellt werden. Sinnvoll ist ein Abstand von 2 oder 3 Punkten. - thr_angle: loop through complete contour and find all locations with concavity angle < anglethr or convexity angle > anglethr - maxlen: >=0: maximale Länge der gezeichneten Trennungslinie,d.h. maximaler Abstand zweier konkaven Konturkrümmungen die als zusammengehörig erkannt werden können. <0: don't change. (- color: Farbe zum Zeichnen der Trennungslinien. Only interactively available)
contour_check	4, B (IM)	type, single_list, obj_count, extremity_count: integer; Verschiedene Kontour-Untersuchungen: Konkavitäten oder Konvexitäten. Achtung: das "SaveFreemanFlag muß zu 1 gesetzt sein! type: 0: search concavities, 1: search convexities single_list: 0|1: use single object or object in list obj_count: 0..n: object number in list extremity_count: 0..m Rückgabewerte: IVAR[10] = m IVAR[11] = Center of Extremity.x IVAR[12] = Center of Extremity.y IVAR[13] = x-Dist IVAR[14] = y-Dist IVAR[15] = x+Dist IVAR[16] = y+Dist IVAR[17] = area in pixel, <0 => concave, >0 => convex IVAR[18] = angle: <180: concave, >180: convex IVAR[19] = run-length: in Freeman-Points from init (top-left start point)
dealloc_objects	1, B(IM),	free_contourbuffer: integer; Freigeben der System-Objektliste und bei "free_contourbuffer=1" auch Freigeben des ein Bit/Pixel tiefen Konturspeichers.
plot_objects	4, B/I	inter, type, buf, col: integer; Darstellung gefundener Objekte als Konturen oder gefüllte Objekte. Wenn col<0, wird nur invertiert. buf gibt den Bildspeicher für gefüllte Objekte an und wird bei inter=1 eingefragt. type = 0: plot all overlay buffer contours

		type = 1: fill contours,use buf, use col, inter=-1=all, 0...n-1=select object to fill type = 2:seed object, use col,inter=-1=all, 0...n-1=select object to seed. (Beim "Seeden" werden die Flächenschwerpunkte (Centers of Gravity) mit weißem Punkt markiert. type =3...7: Contour-Plot, use col, inter=-1=all, 0...n-1=select object to plot. type==3: rect, 4:smallest_rect, 5:smallest_ellipse, 6: equivalent_ellipse, 7: freeman_contour
show_objects	1, I	single: integer; Anzeigen der Objekte einzeln nacheinander (single=1) oder alle gleichzeitig.
object_hist	3, B	type, filteriter, roi_min_max: integer; Berechnen eines Histogrammes über die folgenden Objektparameter: type = 0: RectArea (Box) type = 1: Pixel | Subpixel Area type = 2: Width type = 3: Height type = 4: Area/Box type = 5: Ellipse Dmin type = 6: Ellipse Dmax type = 7: Ellipse Dmax/Dmin type = 8: Ellipse Angle type = 9: Ellipse Area type = 10: Roundness type = 11: Gray Mean Value type = 12: Gray Sigma type = 13: Xcg type = 14:Ycg filteriter = 0..n roi_min_max = 0|1: Bei 1 werden die ROI-Grenzen für min/max benutzt (nur für Xcg und Ycg) Return Werte: FVAR[10] = Rho FVAR[15] = min FVAR[11] = a FVAR[16] = max FVAR[12] = b FVAR[17] = aver FVAR[13] = c FVAR[18] = variance FVAR[14] = d FVAR[19] = entropy
save_objhist	1, B/I	interactive: integer; Abspeichern des Objekt-Histogrammes
contour	3, B	nicht implementiert: dies könnte eine sehr schnelle Kontur-Such-Methode sein...

Statistische Vermessung (Manual Kapitel 8.2.):

area_hist	2, B / ID	interactive, exclude_color: integer; Berechnung des Flächenhistogrammes, interaktiv oder im Batch-Mode. Die Pixelanzahl, normalerweise Breite*Höhe der ROI, wird in IVAR[12] zurückgegeben. Hierzu muß exclude_color immer gleich -1 sein. Wenn exclude_color in [0..255] ist, wird der Grauwert (Pixelwert) exclude_color für die Histogrammberechnung nicht berücksichtigt. Hierdurch wird das Histogramm über eine kleinere Fläche berechnet, in IVAR[12] steht dann die tatsächliche Fläche. ReturnValue immer 1. IVAR[10] = Gray_min IVAR[11] = Gray_max FVAR[10] = Gray_mean FVAR[11] = Gray_variance FVAR[12] = Gray_Entropy
save_areahist	1, B/I	interactive: integer; Abspeichern der Histogrammwerte eines Flächenhistogrammes in einen File. Wenn ‘‘interactive’’ zu 1 gesetzt ist, so wird ein Filename interaktiv eingefragt. Andernfalls wird ein vorhandener zum Schreiben geöffneter Datenfile oder MACRO-File benutzt.
get_hist	1, B	value: integer; Abfrage von mit den obigen Funktionen erzeugten Histogrammwerten: value in [0..255], der Tabellenwert wird in FVAR[10] zurückgegeben. Die Berechnung der Tabelle (z.B. mit area_hist oder line_hist) sollte direkt vor dem "get_hist"-Aufruf stehen, da auch andere Funktionen die Histogrammtabelle benutzen und ändern können.
line_hist	4, B (ID)	interactive, imagebuf, linewidth, freqhamming: integer; Calculate histogram of line and show line luminance or line histogram or line frequency spectrum. Hierzu muß zuvor eine Linie mit den Fadenkreuzen gesetzt werden. Wenn dies nicht interaktiv geschehen soll, so können im Macro über die Funktion "set_two_cross 4,B" zwei Fadenkreuze gesetzt werden.. Die Breite der Linie kann mit dem Parameter "linewidth" im Bereich [1..256] gesteuert werden. Hierbei werden dann über die Linienbreite die eingestellte Anzahl von Pixeln gemittelt. Wenn "freqhamming" auf 1 eingestellt wird, so wird zur Fouriertransformation vorher ein Hamming-Fensterfilter angewandt. Der benutzte Bildspeicher wird mit imagebuf angegeben, -1 entspricht dem Videospeicher. IVAR[10] = Gray_min IVAR[11] = Gray_max, IVAR[12]=Line.Length (Max(dx,dy) FVAR[10] = Gray_mean FVAR[11] = Gray_variance FVAR[12] = Gray_Entropy FVAR[13] = Crosshair_Distance (Linelength) FVAR[14] = Line_Angle FVAR[15] = Line Frequency Length (Length of FFT-line in 2**m <= Linelength) FVAR[16] = 1st Mode (Maximum of Power Spectrum after 1st Minimum, determined using polynom or gauss fit) Rückgabe: FVAR[17] = Line.HmaxValue, Value of dominant harmonic.
save_linehist	1, B/I	interactive: integer; Abspeichern der Histogrammwerte eines Linienhistogrammes in einen File. Wenn ‘‘interactive’’ zu 1 gesetzt ist, so wird ein Filename interaktiv eingefragt. Andernfalls wird ein vorhandener zum Schreiben geöffneter Datenfile oder MACRO-File benutzt.
line_data	3, B,	n, lum_hist_freq, dealloc: integer; Abholen der Linien-Daten nach folgendem Schema: **lum_hist_freq = 0:** Luminanzdaten der gewählten Linie n = -1: draw line data to image buffer, set ROI to ((>linelength)*256). This feature is implemented for luminance only! n: [0..MAX(x-Linelength,y-Linelength)] n=0 liegt am Fadenkreuz Nr. 1, d. h. am durchgezogenen Fadenkreuz. Luminanz-Daten werden in IVAR[10] zurückgegeben. Außerdem werden Daten der Regressionsanalyse abgeholt: IVAR[11] = Result: 1 = in Ordnung FVAR[10] = Rho FVAR[11] = a FVAR[12] = b FVAR[13] = c FVAR[14] = d mit y=ax+b und x=cy+d **lum_hist_freq = 1:** Häufigkeitsverteilung der Grauwerte:

		n: [0..255] Pixelwert / Grauwert. Gesamthäufigkeit ist 1.0. Daten werden in FVAR[10] zurückgegeben. **lum_hist_freq = 2:** Frequenz-Daten der am 1.Fadenkreuz beginnenden Teillinie der Länge 2**m <= Linelength. n: Harmonische [0..Line Frequency Length]. Null ist der Gleichanteil (DC-Anteil). Die Daten werden folgendermaßen zurückgegeben: FVAR[10] = Realteil FVAR[11] = Imaginärteil FVAR[12] = Amplitude (Power) FVAR[13] = Phase dealloc = 1: Freigeben der Buffer, die intern zur Berechnung der Frequenzdaten bereitgestellt wurden (für Realteil, Imaginärteil, Amplituden- und Phasenspektrum)
histo_filter	4, B	type, iter, cyclic, exclude: integer; Histogramm-Filter: Filterung eines 1-dimensionalen Histogramm-Vektors über 256 Grauwerte. Ein solcher Vektor wird von allen internen Histogramm-Funktionen erzeugt. type = 1: recursive Filter: starker Filter mit Mittelung des aktuellen Pixels und des nächsten Nachbarn vor und zurücklaufend. type = 2: Gauss-Filter: schwacher Filter mit [1,2,1] Maske iter = 0..n: Anzahl der Filteriterationen cyclic = 0|1: Bei cyclic=1 wird zyklisch gefiltert, Grauwert 255 hängt an Grauwert 0 exclude = -1, 0..255: Einen Grauwert vom Filtern ausschließen, -1: nicht wirksam
histo_getmax	4, B/I	interactive, max_all_single, cyclic, exclude: integer: Suchen von Maxima im Grauwerthistogramm: interactive = 1: Interaktive Dialogbox interactive = 0: Ausgabe über Macrovariable IVAR[10] = Tab_Count, d. h. Anzahl der lokalen Maxima im Bereich [0..255] IVAR[11] = Number of Gray Levels in Table (255 bei exclude!=-1, sonst 256) max_all_single = 0: Suche absolutes Maximum FVAR[10] = Maximum, FVAR[11] = Ort des Maximum [0..255] (eventuell interpoliert) max_all_single = 1: Suche alle lokalen Maxima (von links nach rechts im Histogramm) FVAR[10] = 1.Maximum in %, FVAR[11] = Ort des 1.Maximum [0..255] FVAR[12] = 2.Maximum in %, FVAR[13] = Ort des 2.Maximum [0..255] FVAR[14] = 3.Maximum in %, FVAR[15] = Ort des 3.Maximum [0..255] usw. bis zum 5. Maximum max_all_single = 2: Ausgabe eines lokalen Maximum max_all_single < 0: get selected max, and output sorted table. interactive = Nummer = 0..n (von links nach rechts im Histogramm) FVAR[10] = Maximum[Nummer] in %, FVAR[11] = Ort des Maximum[Nummer] Bei negativem max_all_single: -1|-2|-3: get max-n Wenn cyclic zu 1 gesetzt ist, so wird ein Maximum auch um den Nullpunkt herum gesucht. Mit exclude kann ein bestimmter Wert, z. B. der Nullwert, von der Maximumsuche und auch von der Filterung (s.o.) ausgeschlossen werden.
line_value	1, I	direction: integer; Anzeige der Luminanz einer Linie: Richtung = 1: horizontal, Richtung = 2: vertikale Linie. Dabei wird der Anteil der Linie in der aktuellen ROI benutzt. Return_Value immer 1. Werte für die letzte angewählte Linie: IVAR[10] = Gray_min IVAR[11] = Gray_max FVAR[10] = Gray_mean FVAR[11] = Gray_variance FVAR[12] = Gray_Entropy
pointhist	8, B/I	interactive, init: integer; val, dummy: float; Berechnen eines Punktscharhistogrammes. Im interaktiven Modus werden mit einem Fadenkreuz Pixelwerte eingegeben. Bei init=0 werden die alten Werte aus dem letzten Aufruf der Funktion beibehalten. Ein Pixel wird durch linken Mausklick, die Return-Taste oder die +(plus)-Taste eingegeben. Bei Fehleingabe kann mit der - (minus)-Taste der letzte Pixel gelöscht werden. Die Funktion wird mit der rechten Maustaste oder der ESC-Taste beendet. Bei nichtinteraktiver Funktion werden die floatingpoint-Werte val eingegeben. Bis auf den ersten Aufruf muß also init immer auf 0 gesetzt sein. init = 1: initialisieren aller Werte val: FloatingPoint Zahlenwert FVAR[10] = min

		FVAR[11] = max FVAR[12] = mean FVAR[13] = variance FVAR[14] = std.dev.
single_coocmat	1, ID,	Cooc_2ch_Hist: integer; Berechnung der Coocurrence-Matrix oder eines 2-kanaligen Histogrammes, wenn Cooc_2ch_Hist = 1. Es muß dann das Second_Source-Bild gesetzt sein. Return_Value 1 bei erfolgreicher Berechnung der Cooc-Matrix, -1 bei Speichermangel. Folgende Werte einer Gray Level Coocurrence Matrix werden zurückgegeben, wenn die Features in der Coocurrence-Dialogbox angewählt werden: FVAR[10] = Angular Second Moment FVAR[11] = Contrast FVAR[12] = Correlation FVAR[13] = Inverse Difference Moment FVAR[14] = Entropy FVAR[15] = Variance FVAR[16] = Cluster Prominence FVAR[17] = Cluster Shade FVAR[18] = Inertia FVAR[19] = Diagonal Moment
cooc_matrix_feat	1, B/ID,	interactive: integer; Select feature for coocurrence matrix filter
multi_coocmat	0, ID,	kein Parameter; Filterung des Bildes mit einem Coocurrence-Matrix-Verfahren. Return_Value -1, da nicht implementiert.
regr_analysis	8, B	type, n: integer; x,y: float; Regressionsanalyse eingegebener Wertepaare. type = -1: Löschen der Liste und Freigeben des Speicherbereiches type = 0: Eingeben der Wertepaare x,y type = 1: Liste definieren mit der Länge n type = 2: Berechnung der Regressionskoeffizienten und -geraden Die Werte werden folgendermaßen ans Macro-Programm zurückgegeben: FVAR[10] = rho (Regressionskoeffizient) FVAR[11] = a FVAR[12] = b FVAR[13] = c FVAR[14] = d mit y = ax+b und x=cy+d

Edge Detection (Manual Kapitel 8.4.):

Alte Funktionen:

hough	0, B,	Kein Parameter; Hough-Transformation zur Bestimmung von Kanten und Linien, die auch unterbrochen sein dürfen.
draw_hough_all	3, B,	col, max, clip: integer; Plotten oder Anzeigen aller gefundenen Linien
draw_hough_max	2, B,	col, clip: integer; Plotten der am stärksten ausgeprägten aller gefundener Linien
hough_all_prep	1, B,	vorläufige Funktion zur Aufbereitung der Daten im Hough-Raum
transfer_hough	0, B	kein Parameter; Transfer der Daten des Hough-Raumes in einen Bildspeicher

Neue Funktionen:

hough_advanced	8, B/I	type, filteriter: integer; linethickness, perimeterpart: float; Advanced function for Hough transform for lines, circles, ellipses
hough_params	4, B	type, p1, p2, p3: integer; Parameter für die Hough-Transformation: type = 2: circle, p1 = minrad, p2 = maxrad, p3 = diff type = 3: ellipse, p1 = minrad, p2 = maxrad, p3 = diff type = 4: common param input, p1 = neighborhood, p2 = minhist, p3 = detection roi type = 5: common param input, p1 = display, p2 = number to display, p3 = histo window size
hough_data	1, B	n: integer; Ausgabe der Hough-Daten IVAR[10] = -1: no data, 0: upwrite, 1: line, 2: circle, 3: ellipse
hough_draw	1, B/I	color: integer; Graphische Darstellung der gefundenen Elemente durch die Hough-Transformation

Funktionen zur Vorbereitung der Kantenerkennung (siehe lineare und nichtlineare Filter):

marrhil	2, B	lapl, grad: integer; Marr-Hildreth-Operator zur Kantenextraktion
canny	4, B	type, low_thresh, high_thresh, edge_buffer: integer; Canny-Edge-Operator zur Kantenextraktion
zerocross	1, B	frame: integer; Bestimmung der Nulldurchgänge im Bild (Zero Crossings).
edge_trace	4, B/I	threshold, overlay, xstart, ystart: integer; Tracen von Kanten

Object Classification (Manual Kapitel 8.5.):

class_mindist 1, B/ID

interactive: integer; Objektklassifikation mit dem Minimum-Distance- oder dem Parallel-Epiped-Verfahren.

set_class_inp 4, I/B

inter, dim, channel, cyclic: integer; Input der Bildkanäle für eine Klassifikation
inter = 1: interaktive Eingabe, restliche Parameter nicht benutzt
inter = 0: nichtinteraktive Eingabe: 1 < dim < 4: Anzahl der Bildkanäle (Merkmale)
channel: Bitweise Angabe der Kanäle (Merkmale)
Bit 0: Bildspeicher 1
Bit 1: Bildspeicher 2 usw.
Z. B. bedeutet "5" die Bildspeicher 1 und 3, "15" bedeutet Bildspeicher 1,2,3,4. Es werden nur [dim] Bits vom Wert in [channels] eingelesen und benutzt, beginnend mit dem niedrigwertigsten Bit
cyclic: Bitweise Angabe des zyklischen Bildspeichers (z. Zt. nur einer erlaubt). Gesetztes Bit 15 (d.h. plus dezimal 32768) besagt, daß Pixelwerte von "0" bei den zyklischen Kanälen nicht berücksichtigt werden, wie es z. B. beim HUE-Kanal erforderlich ist, bei dem "0" "keine" Farbe bedeutet.
Wird inter = -1 gesetzt, so werden folgende Werte in die MACRO-Variable übertragen:
IVAR[10] = mode (s.u.) IVAR[11] = dim
IVAR[12] = max number of clusters IVAR[13] = actual number of clusters
IVAR[14] = threshold for euclidian distance (for mode=1,2)
IVAR[15] = global multiplicative distance factor (for mode=3)
IVAR[16] = channel (bitwise)
IVAR[17] = cyclic (bitwise, bit 15=1:don´t count "0")

set_obj_mode 1, B

mode: integer; Klassifikations-Modus:
mode = 1: Automatische Euclidian Minimum Distance Class.
mode = 2: Trained Euclidian Minimum Distance Classification
mode = 3: Trained Parallel Epiped Min. Distance Classification
mode = -1: nur Abfrage des Modus.
IVAR[10] = mode, wird immer gesetzt.

class_saveload 2, I

saveload, interactive: integer; Abspeichern und Laden der Klassifikationsparameter. Hierbei ist z. Zt. die Filenameneingabe immer interaktiv. Die Daten werden in einen Binär-File mit der Extension *.OCP gespeichert. Alle Werte sind 32 Bit Integer.
mode [1,2,3], dimension, maximum number of cluster (20)
actual number of cluster
threshold (euclidian distance, used only with mode=1,2)
multiplicative distance factor (parallel epiped, mode=3)
channel-buf1, channel-buf2, channel-buf3, channel-buf4
cyclic-flag1, cyclic-flag2, cyclic-flag3, cyclic-flag4
(0: not cyclic, 1: cyclic, 2: cyclic, don´t count "0")
4 * 20 * flag: cluster in cyclic channel is at 255/0-border
4 * 20 * center of cluster
4 * 20 * max. distance of cluster (mode=3, par. epiped)
20 * threshold (not used)
20 * zz (internal value for cluster distances)
ReturnValue ist 1 bei erfolgreicher Ausführung, 0 bei falscher (leerer) Filenameneingabe, und -1 bei Fehler (File not Found oder Schreib/Lesefehler)

train_mindist 2, I

center, distance: integer; Trainieren der Klassifikationsparameter:
center = 1: Zentrum = (Min + Max) / 2, wobei MIn und Max die jeweils kleinsten und größten Werte im jeweiligen Merkmalsspeicher (channel) sind. Achtung: Fehlermöglichkeit durch falsch angeklickte Punkte.
center = 2: Als Zentrum wird der Mittelwert aller eingegebenen Trainingspunkte bestimmt. Die Angabe des Abstandes ist nur für die Parallel-Epiped-Methode sinnvoll, für die Euclidian-Distance-Methode werden kreis- oder kugelförmige Bereiche mit wählbarer Größe genommen.
distance = 1: Abstand vom Zentrum = (Max - Min) / 2, wobei MIn und Max die jeweils kleinsten und größten Werte im jeweiligen Merkmalsspeicher sind. Achtung: Fehlermöglichkeit durch falsch angeklickte Punkte.
distance = 2: Abstand ist die Standard-Abweichung der eingegebenen Trainingswerte. Hier kann später in der Dialogbox ein multiplikativer Faktor gewählt werden, sodaß man z. B. Punkte bis zur 2-fachen Standardabweichung zulassen kann.

Stereometrische Vermessung (Kapitel 8.7. und 14. und Zusatzmodul):

Stereo-Vermessung:

stereo_vector	2, I	lines, originvisible: integer; Stereometrische Auswertung von zwei unter 90 Grad aufgenommenen Bildern. Return_Value immer 1.
stereo_h_v	1, B,	Horizontal_Vertikal: integer; Anordnung der Bilder horizontal nebeneinander (0) oder vertikal übereinander (1) für die Funktion stereo_vector(). Return_Value immer 1.

Stereo-Handling und Stereo-Display:

stereo_buffer	4, B/I	inter, left, right, mode: integer; Angabe der Bildspeicher und weiterer Parameter für Stereoanwendungen, siehe Stereo-Zusatzmodul inter = -1: Get Variables: IVAR[10]=mode, IVAR[11]=left buffer, IVAR[12]=right buffer, FVAR[10]=f1 FVAR[14]=f2 FVAR[11]=kappa1 FVAR[15]=kappa2 FVAR[12]=x1 FVAR[16]=x2 FVAR[13]=y1 FVAR[17]=y2 inter = 0: Angabe der Stereo-Bildspeicher mit 'left' und 'right' und Einschalten des Stereo-Modus mit mode = 1. inter = 1: Interaktive Dialogbox inter = 2..15: Eingabe weiterer Parameter über die Variablen "left" und "right" für die linke und rechte Kamera, z. B. für die Linsenentzerrung im Stereo-Modus mit zwei Kameras. Genauere Beschreibung dieser Werte findet man bei der einfachen Linsenverzerrung (Camera Calibration, im Haupthandbuch). Set Camera Calibration Variables: (use left/right for both camera views 1 and 2) 2: Ncx 3: Nfx 4: dx 5: dy 6: dpx 7: dpy 8: cx 9: cy 10: sx 11: f 12: kappa 13: p1 14: p2 15: ipol (Interpolationsart: 0|1|2 = keine|lin-ipol|spline) Mit mode=1 werden die Stereo-Funktionen zugelassen. Dies ist für die Positionsbestimmung normalerweise nicht notwendig. Werte, die negativ eingegeben werden, werden nicht geändert.
stereo_display	2, I	left, right: integer; Stereo Display mit LCD-Shutterbrille. Über die Variable Vidstrm.FPR kann die Frame Rate in 1/s eingestellt werden. Die Darstellungslänge kann über Stereo.field in Sekunden/100 eingestellt werden, siehe Stereo-Zusatzmodul

Pattern Recognition (Manual Kapitel 8.6.):

match_regress	4, B	ref_roi, ref_buf, stepsize, findinvert: integer; Mustererkennung über die Regressionsanalyse. Hierbei wird das gesuchte Referenzmuster durch die Referenz-ROI und den Referenzbildspeicher gegeben. Das Muster wird dann in der kompletten aktuellen ROI gesucht. Die Referenz-ROI muß aus Geschwindigkeitsgründen möglichst klein sein, sollte exakt das gesuchte Muster umschließen. Bei einer Schrittweite von 2 (stepsize=2) geht das ganze 4mal so schnell, dafür kann es passieren, daß sehr kleine Strukturen nicht oder nur ungenau gefunden werden, daher nur als Vorschau benutzen. Mit findinvert=1 können auch grauwertinvertierte Muster gefunden werden.
match_xcorr	4, B/IM	ref_roi, ref_buf, hamming, twopass: integer: Mustererkennung mit der Kreuzkorrelation. Es können alle Stellen in der aktuellen ROI hervorgehoben werden, an denen eine mehr oder weniger gute Übereinstimmung mit einem Referenzmuster auftritt. Da die Fouriertransformation benutzt wird, müssen aktuelle und Referenz-ROI quadratisch und von der Größe 2**n sein. Die Referenz-ROI darf gleich der aktuellen ROI sein, sie kann jedoch auch kleiner sein. Der Referenz-Bildspeicher darf gleich dem aktuellen Bildspeicher sein. Das gesuchte Muster muß sich genau in der Mitte der Referenz-ROI (ref_roi) befinden, sonst wird eine zusätzliche Verschiebung miteingerechnet. Wenn nicht das Auftreten von Mustern an mehreren Stellen im aktuellen Bild bestimmt werden soll, sondern eine Verschiebung vom aktuellen Bild gegenüber dem Referenzbild, so müssen beide ROI´s gleich groß sein. In diesem Fall kann die Genauigkeit der Auswertung um beinahe eine Größenordnung gesteigert werden, indem ein zweiter Pass (twopass=1) durchlaufen wird, bei dem die aktuelle ROI um die im ersten Pass gefundene Verschiebung vorverschoben wird. Hierdurch stimmen die Inhalte der ROI´s besser überein und der Korrelationspeak wird viel steiler. Die ROI´s werden wieder richtig restauriert. Zum Suchen der Maxima im Korrelationsspektrum werden zwei Konstante benötigt, sigma_max und sigma_min. Diese können mit der Funktion findmax(0,-1,sigma_max,sigma_min) gesetzt werden (siehe unten). Die drei stärksten Korrelationsmaxima werden in den MACRO-Variable abgelegt: FVAR[10] = dx des größten Korrelationspeaks FVAR[11] = dy des größten Korrelationspeaks FVAR[12] = dx des zweitgrößten Korrelationspeaks FVAR[13] = dy des zweitgrößten Korrelationspeaks FVAR[14] = dx des drittgrößten Korrelationspeaks FVAR[15] = dy des drittgrößten Korrelationspeaks FVAR[19] = letztes intern benutztes sigma (>=sigma_min) IVAR[10] = Normalisierter Corr.Coeff. des absoluten Korrelationspeaks IVAR[11] = Normalisierter Corr.Coeff. des zweitgrößten Peaks, jeweils [0..100] IVAR[12] = Normalisierter Corr.Coeff. des drittgrößten Peaks, 100 entspricht 1.0 Bei twopass=-1 oder -2 wird der Realteil des Bildes in den aktuellen Bildspeicher kopiert, bei twopass=-1 mit schwarzem Hintergrund, bei twopass=-2 mit dem Average als Hintergrund. In diesem Bild können dann lokale Maxima gesucht werden.
match_rotinv	4, B	refbuf, refroi, hamming, dummy: integer; Pattern Matching mit einem rotationsinvarianten und größeninvarianten Ansatz. refbuf = 1..n, der Referenzbildspeicher refroi = 0|1|2, die Referenz-ROI Hamming = 0|1: Ein Hamming-Filter kann für die Ausführung der FFT Vorteile bringen. Returnwerte: IVAR[10][11][12] sind die Korrelationskoeffizienten der drei wahrscheinlichsten Verschiebungen, die x,y-Werte der drei wahrscheinlichsten Verschiebungen findet man in FVAR[10][11], FVAR[12][13], FVAR[14][15]. Die Winkel stehen in FVAR[16][17][18]. Achtung: wegen der Frequency Domain besteht eine 180 Grad Ambiguity!
findmax	8, B/I	interactive, auto_xcorr; integer; sigma_max, sigma_min: float; Aufsuchen der 3 Grauwert-Maxima innerhalb einer ROI. Wenn die Funktion interaktiv aufgerufen wird (inter=1), so werden die drei gefundenen Maxima durch Kreise angezeigt und ihre Lage mit dx und dy-Verschiebung vom Mittelpunkt der ROI (und eventuell auch mit Radius und Winkel) angegeben. Für Bildinhalte mit beliebig liegenden Maxima oder für Kreuzkorrelationsergebnisse muß (auto_xcorr = 1) sein, da sonst im Mittelpunkt ein Autokorrelationspeak oder der DC-Anteil des FFT-Powerspektrums erwartet wird. Für Autokorrelation oder FFT-Powerspektrum (auto_xcorr = 0) setzen! Mit (auto_xcorr = -1) werden lediglich die internen Werte für sigma_max und sigma_min besetzt, damit sie

auch von der match_xcorr-Funktion verwendet werden können. Mit (auto_xcorr = -2) können die zuvor gefundenen Werte zur Inspektion ausgegeben werden (und ebenfalls auf die Macro-Variable geschrieben werden).
sigma_max ist ein Wert für den Abstand zweier Maxima, die noch als getrennt gesehen werden sollen. Standardwert ist 2.8, bei sehr diffusen und flachen Maxima ist es sinnvoll, den Wert bis auf ca 2.0 oder im Extremfall sogar auf 1.0 zu verringern. Dabei wird die Suche jedoch recht unsicher... Wenn vor Funktionsausführung, z. B. bei Benutzung der match_xcorr-Funktion, die intern die findmax-Funktion aufruft, nicht bekannt ist, ob die Maxima scharf oder diffus sein werden, kann man die Eingabe dieser Größe variabel gestalten: In sigma_max wird der Startwert eingegeben, in sigma_min der kleinste erlaubte Wert. Das System beginnt nun, mit dem Startwert zu suchen. Wenn keine ausgezeichneten Maxima gefunden wurden, wird der sigma-Wert um 0.05 erniedrigt und eine neue Suche gestartet. Dies wird wiederholt, bis entweder Maxima gefunden werden oder die sigma_min-Grenze erreicht wurde. Im letzten Fall wird -1 zurückgegeben, sonst 1. Der letzte benutzte sigma-Wert wird ebenfalls übergeben, so daß man neue Untersuchungen ähnlicher Bilder durch bessere sigma_max-Eingabe schneller ausführen kann.
Die Art der Maximasuche hängt noch von weiteren Parametern ab, die über die Funktion meas_control eingegeben werden können, z. B. aver_cross und gauss_polynom
Meßwerte werden wie folgt zurückgegeben:
FVAR[10] = dx des größten Korrelationspeaks
FVAR[11] = dy des größten Korrelationspeaks
FVAR[12] = dx des zweitgrößten Korrelationspeaks
FVAR[13] = dy des zweitgrößten Korrelationspeaks
FVAR[14] = dx des drittgrößten Korrelationspeaks
FVAR[15] = dy des drittgrößten Korrelationspeaks
FVAR[19] = letztes benutztes sigma (>=sigma_min)
IVAR[10] = Pixelwert des absoluten Korrelationspeaks
IVAR[11] = Pixelwert des zweitgrößten Peaks
IVAR[12] = Pixelwert des drittgrößten Peaks

Particle Image Velocimetry / Particle Tracking / Time-Resolved-PIV:

(Manual Kapitel 8.9. und 14. und im PIV/PTV-Zusatzmodul)

1: Particle Image Velocimetry:

piv_correlation	8, B (IM)	auto_cross, twopass: integer; hamming, sigma: float; Autokorrelation mit einem doppelt belichteten Partikelbild oder Kreuzkorrelation mit zwei zeitlich aufeinanderfolgenden Partikelbildern. Bei Kreuzkorrelation 2. Durchlauf möglich, der das Ergebnis wesentlich genauer macht und auch die Sicherheit erhöht.
rt_trpiv	4, B / I	type, param1, param2, param3: integer; Echtzeit-PIV, Time-Resolved-Particle-Image-Velocimetry: type = 2 (continuous TRPIV) oder 3 (PIV Correlation mit den aktuellen Bildern im 3D-Sequenz-Speicher. param1, param2, param3 noch nicht genutzt. Zur Parametereinstellung werden die Funktionen rt_param(), rt_param2(), piv_param() und piv_param2() eingesetzt.
rt_param2	4, B / I	type, param1, param2, param3: integer; Setzen vieler Parameter für TRPIV-Funktion und auch für andere Echtzeit-Funktionen. Zur Beschreibung siehe Real-Time-2D/3D-Position-Tracking.
piv_param	4, B	roisize, stepx, stepy, subpixel: integer; Parametereingabe für die Korrelation: Größe des Korrelationsfensters, Schrittweite in x- und y-Richtung, und Bestimmung in Subpixelgenauigkeit.
piv_param2	8, B	inter, refimage_type: integer; min, max: float; Zusätzliche Parametereingabe für die Korrelation. inter=1: interaktive Eingabe aller Parameter refimage_type = -1: PIV Parameter Dialogbox, but do not allow PIV-Correlation! refimage_type = 0: PIV Parameter Dialogbox refimage_type = 1: PIV Rejectioncriteria Dialogbox inter=0: refimage = Referenz-Bildspeicher, min=Zeitabstand zweier PIV-Bilder in [ms]. inter = 2: Rejection Criteria: refimage_type = 0: kein Mittelwertbilden von zurückgewiesenen Null-Vektoren refimage_type = 1: Alle Null-Vektoren (dies sind normalerweise zurückgewiesene Vektoren) werden zum Mittelwert der direkten (von Null verschiedenen) Nachbarn gesetzt. refimage_type = 2: Keine Anwendung der Rejection Criteria refimage_type = 3: Die eingegebenen Rejection Criteria werden benutzt refimage_type = 4: Minimum und Maximum der zugelassenen Verschiebung (Geschwindigkeit) werden in den Parametern min/max übergeben, z. Zt. in Pixeleinheiten, d. h. Verschiebungen in Bildschirmkoordinaten refimage_type = 5: Eingabe eines zurückgewiesenen Winkelbereiches. Minimaler und maximaler Winkel des zurückgewiesenenen Bereiches werden in min/max übergeben. Hierbei sind Werte im Bereich [-360.0...360.0] Grad zugelassen. Bei der Datenangabe muß gelten: min < max, sonst wird keine Aktion ausgeführt. refimage_type = 6: Eingabe des kleinsten zugelassenen Korrelationskoeffizienten für Rejection, if corr.coeff. < min refimage_type = 7: Benutze lokale Feld-Nummer für Statistik, z. B. Average der Geschwindigkeiten über mehrere Vektorfelder (GlobalFieldNr=0). Null-Vektoren werden hierbei nicht berücksichtigt. refimage_type = 8: Benutze globale Feld-Nummer (GlobalFieldNr=1). Alle Vektoren werden für die Statistik benutzt, auch Null-Vektoren.
piv_vorticity	1, B	nr: integer; Berechnung des Vorticity-Feldes des Vektorfeldes[nr]. Die Ergebnis-Wirbelstärken werden im Vorticity-Feld abgelegt und können mit der Funktion "piv_surface (-1,-1)" angezeigt werden.
piv_getvect	4, B/I	interactive, y, x, nr: integer; Abfragen der Vektordaten (x,y) von Vektorfeld Feld[nr]. Wenn "interactive" gleich 1 ist, wird eine Dialogbox ausgegeben, in der alle Vektordaten dargestellt sind. Nur bei "interactive" = 0 können Einzelvektoren abgefragt werden. Die Daten werden wie folgt abgelegt:

FVAR[10] = x FVAR[12] = dx
FVAR[11] = y FVAR[13] = dy
FVAR[14] = choice FVAR[15] = corr.-coeff.
Hierbei werden die Daten kalibriert im Systemkoordinatensystem ausgegeben. Wenn nicht kalibriert wurde, wird in Pixeleinheiten ausgegeben..

piv_drawvect 4, B(I) col, line_arrow, nr, edit: integer; Anzeige, Plotten oder Editieren der Vektordaten aus Feld[nr] mit der Farbe "col" im Grauwertebereich [-2, -1,0..255], wobei bei col=-1 invertiert dargestellt wird. Bei col=-2 werden die Vektoren in einem Grauwert dargestellt, der ihrer Länge entspricht. Dieser Grauwert kann auch kalibriert werden. Hierzu wird ein kalibrierter Farbbalken dargestellt. Die Kalibrierung des Farbbalkens geschieht normalerweise automatisch, kann jedoch mit der Funktion color_bar_cal auch festgelegt werden. Dabei können die Vektoren mit oder ohne Vektorköpfen dargestellt werden, um die Zweideutigkeit bei der Autokorrelation zu verdeutlichen (line_arrow=1 zeichnet nur Linienstücke, z.B. für eine zweideutige Auto-Correlation, line_arrow=2 zeichnet Vektorpfeile). Bei line_arrow<0 werden alle Vektoren gleich lang dargestellt, und zwar mit der Länge, die durch den pfactor der Funktion piv_plotfac festgelegt wurde. In diesem Fall sollte zur Darstellung der Geschwindigkeit mit Graustufen col auf -2 gesetzt werden. In diesem Fall muß beachtet werden, daß keine Minimalwerte für den kalibrierten Farbbalken von kleiner als 0.0 eingegeben werden können. Werte <0.0 werden zu 0.0 gesetzt. Dies macht Sinn, da die Vektorlänge kalibriert wird, negative Längen sind nicht definiert.Wenn edit auf 1 gesetzt wird, wird das gesamte Vektorfeld zum Editieren freigegeben. Durch Anklicken eines Vektors mit der linken Maustaste werden die 3 wahrscheinlichen Vektoren durchgescant und dargestellt (0,1,2,0,1,2,...). Nach rechter Maustaste wird der aktuelle Stand in das Vektorfeld[nr] übernommen. Mit shift und linker Maustaste können Vektoren auch über eine Gauss-Filter-Konvolution den Nachbarvektoren angeglichen werden. Shift und linke Maustaste kann dazu mehrfach hintereinander angewandt werden.Wen edit auf -1 gesetzt wird, dann werden die Vektoren nur invertiert, nicht gelöscht und es wird auch keine Skalierungs-Box angezeigt.

piv_saveload 4, B/I saveload, nr, interactive, bits: integer; Abspeichern oder Hereinladen der Vektordaten des Vektorfeldes mit der Feldnummer "nr" auf einen oder von einem Datenfile.
saveload = 0: Abspeichern der Vektordaten. Zuerst wird geprüft, ob ein Datenfile zum Schreiben im MACRO-Programm geöffnet ist. Wenn ja, so wird dieser benutzt. Wenn nein, wird geprüft, ob ein Systemdatenfile geöffnet ist. In diesem Falle wird dieser benutzt. Wenn kein Meßdatenfile geöffnet ist, so wird interaktiv ein Filename eingefragt und ein ASCII-File geöffnet und nach der Abspeicherung wieder geschlossen. Wenn vorher schon ein File geöffnet war, so wird dieser nicht geschlossen. Es wird Feld "nr" abgespeichert. Hierbei wird eine System-Kalibrierung mit berücksichtigt. Es wird dann allerdings immer in Meter [m] umgerechnet und gespeichert. Außerdem geschieht eine Umrechnung in das definierte System-Koordinatensystem. Die Daten werden im folgenden Format abgespeichert:
20 Kalibrierungs-Header System-Unit ist [pixel] oder [m]
Cal_x, Cal_y, Origin_x, Origin_y, Rotation, Y_Down
22 ROI Coordinates Header
x1, y1, x2, y2
24 Geometry Parameters
dx, dy, angle, stpx, stpy, 2pass, pol/gauss, hamm, best, opt.axis
25 x,y-Feldgröße-Header
Anzahl_x Anzahl_y
26 Daten Header
x1 y1 dx1 dy1
x2 y2 dx2 dy2
... usw.
Das Abspeichern der Kalibrierung, der ROI-Koordinaten und der geometrischen Parameter kann über die "bits"-Eingabe ein- oder aus-geschaltet werden. Wenn keine Daten vorhanden sind, wird folgender String abgespeichert: *27* No PIV/FCM data
saveload = 1: Laden eines Vektorfeldes in Feldnummer nr. Wenn interactive==0, so wird ein vorhandener MACRO-Datenfile, der zum Lesen geöffnet war, benutzt. Wenn kein Datenfile geöffnet ist, so wird die Funktion beendet und 0 zurückgegeben. In der Regel wird beim Einlesen das gesamte Vektordatenfeld gelöscht und neu definiert, da sich ja andere Feldgrenzen ergeben könnten. Falls dies nicht erwünscht ist und in ein vorhandenen Feld geladen werden soll, so kann dies mit interactive==(-1) getan werden. Wenn dabei das Vektorfeld nicht mit dem schon definierten übereinstimmt, so wird das neue Feld Vektor für Vektor in das vorhandene Feld eingelesen und dabei eventuell abgeschnitten. Offene Files werden nicht geschlossen. Bei interactive==1 wird ein

Filename eingefragt und der File hinterher wieder geschlossen. Beim Einlesen werden alle Daten mit dem im File-Header gespeicherten Koordinatensystem wieder in die interne Pixel-Darstellung umgerechnet. Die Header-Einstellung (mit dem Parameter "bits") muß der Einstellung beim Abspeichern der Daten entsprechen. Wenn das Bit0 im Parameter "bits" gesetzt ist, so wird die zugehörige Kalibrierung miteingelesen. Entsprechendes gilt für die ROI-Koordinaten: entweder vorher auf die gleichen Werte einstellen oder beim Abspeichern der Daten mitspeichern (Bit2 von "bits" gesetzt).

Bit4	Bit3	Bit2	Bit1	Bit0
Params	Koeff.	ROI-Daten	alpha,beta,dist	cal_x_y

Beispiele: bits=21: speichere und lade Kalibrierung, ROI-Daten, PIV-Parameter. bits=5: speichere und lade Kalibrierung und die ROI-Daten. Bit1 und Bit3 werden nur vom FCM-Modul benutzt.
Wichtig: bits muß den gleichen Wert für Abspeichern und Einlesen der Daten haben, sonst können die Daten nicht wieder eingelesen werden! Beim Einlesen von Daten muß außerdem das Header-Flag im Measurement-Control-Menü die gleiche Einstellung haben wie beim Abspeichern. Dies kann über die MACRO-Funktion data_params(Headers,DecPoint,Delimiter) eingestellt werden.

piv_surface 2, B

field_n, x_y_xy: integer; Anzeige der Verschiebevektoren oder Vorticity-Felder als interpoliertes Grauwertbild.
field_n ist das angezeigte Vektorfeld [0..2]. Bei field_n==-1 wird das Vorticity-Feld angezeigt. Die (x_y_xy-Variable hat in diesem Fall keinen Einfluß.)
x_y_xy=1|2|3: Einstellung der angezeigten Größe. Es kann entweder der x-Wert der Vektoren (vorzeichenbehaftet) oder der y-Wert (vorzeichenbehaftet) oder auch der Gesamtbetrag sqrt(x*x+y*y) dargestellt werden. Bei "-1" keine Änderung.
Nach Beendigung kann mit color_bar ein kalibrierter Farbbalken gezeichnet werden. Die Kalibrierung kann automatisch oder manuell erfolgen, wobei dies vor der Ausführung der Funktion piv_surface mit color_bar_cal vorgenommen werden muß:
autom = 1 automatische Wahl der Parameter zur vollen Ausnutzung des Grauwertebereiches, setzen der internen Variablen als Offset und Gain
autom = 2 automatische Wahl der Parameter zur vollen Ausnutzung des Grauwertebereiches, setzen der internen Variablen als Min und Max
autom = -1 benutze min/max als offset und gain zur Bestimmung der Grauwertverteilung
autom = -2 benutze min und max zur Bestimmung der Grauwertverteilung
autom = 0: keine Änderung der autom-Variable.
In die Macrovariable werden folgende Werte abgelegt:
FVAR[10] = offset FVAR[11] = gain
FVAR[12] = min FVAR[13] = max
min und max entsprechen dabei den Werten für Color-Bar-Min und Max. Die dargestellten Grauwerte können nun auch punktweise kalibriert vermessen werden. Hierzu muß im "Measurement Control"-Menü die Pixelvermessung auf "Gray Level Calibrated" eingestellt werden. Wenn nun im ROI_Edit-Modus Einzelpixel vermessen werden, so wird der kalibrierte Grauwert bezüglich der Min/Max-Werte des Color-Bar angegeben, kalibriert in pixel-Längen oder der eingestellten Längeneinheit.
(z. Zt. noch nicht in Geschwindigkeiten, d.h. es werden nur die Verschiebungen zwischen beiden Bildern in Pixeleinheiten oder in einer kalibrierten Einheit angegeben.)

piv_function 8, B

type, nr: integer; param1, param2: float; Filterung oder Korrektur des Vektorfeldes "nr", z. B. mit einem Gauss-Filter. Es kann eine beliebige Maske angegeben werden (im Arbeitsspeicher ("param1=maskbuf" != 0) zu 255 gesetzte Bereiche), innerhalb derer gefiltert wird.
type = 1: Filtern des Feldes [nr] mit einer Gauss-Maske (1,2,1,2,4,2,1,2,1). Hierbei kann ein kernel für die Konvolution angegeben werden:
param2 = kernel = 1: vertikal (1,2,1)
param2 = kernel = 2: horizontal (1,2,1)
param2 = kernel = 3: 3*3 Nachbarschaft mit (1,2,1,2,4,2,1,2,1)
maskbuf = param1
Wenn maskbuf!=0 ist, so werden nur Vektoren für die Filterung benutzt, an denen das Maskenbild den Wert 255 hat. Das Maskenbild liegt dann im Arbeitsspeicher: abs(maskbuf). Wenn maskbuf<0 ist, so werden zusätzlich noch die gerade noch an das Maskenbild angrenzenden Vektoren benutzt.
type = 2: Vektorkorrektur (best choice), nur für FCM-Zusatzmodul.
type = 3: Korrektur der PIV-Vektoren nach bestimmtem Nachbarschaftskriterium (nicht implementiert)
type = 4: Löschen der Vektoren außerhalb der weißen Maske, wenn das Maskenbild maskbuf = param != 0 ist.

		type = 5: Mittelwertbildung aller Null-Vektoren (z. B. zurückgewiesene Vektoren) aus den direkten, von Null verschiedenen Nachbarn. Auch zur Extrapolation von Vektorfeldern, die mit einem Maskenbild teilweise gelöscht wurden. (param2=kernel wie oben, 1=vert., 2=horiz., 3=3*3- Nachbarschaft) type = 6: Mathematische Operationen: Addition, Subtraktion, das zweite Vektorfeld zur Berechnung wird in der Variable param angegeben. param2 = kernel = 1: Addition: Field[n] := Field[n] + Field[param1] param2 = kernel = 2: Subtraktion: Field[n] := Fiend[n] - Field[param1] param1 = maskbuf=0: Filterung ohne Benutzung eines Maskierungsspeichers. param1 = maskbuf>0: Benutze den Arbeitsspeicher "maskbuf" als Maske. Es wird nur in Bereichen gefiltert, in denen der Maskierungsspeicher zu weiß [255] gesetzt ist. Es ist dabei zu beachten, daß das Vektorfeld dabei pixelorientiert (Vektoranfangspunkte) über den Maskierungsspeicher "gemapped" wird.Die aktuelle ROI muß daher richtig gesetzt sein, d. h. sie sollte exakt die gleiche sein, wie sie bei der Korrelation benutzt wurde. Die zur PIV-Vektor-Berechnung benutzte Systemkalibrierung darf nicht verändert werden.
piv_statistic	1, B	type: integer; Statistische Auswertung mehrerer nacheinander berechneter Vektorfelder. Hierzu müssen die Vektorfelder nacheinander von Disk in das Vektorfeld [0] hereingeladen werden (z.B. mit einem MACRO und piv_saveload) und mit piv_statistic(0) in die Statistik-Variablen addiert werden. Die Statistik-Größen können dann jederzeit in das Vektorfeld[0] geschrieben werden. Als Statistik-Variable werden die internen Vektorfelder [1] und [2] und [vorticity] verwendet. type = -1: Reset Statistik-Felder (dies sind Vektorfelder[1], [2] und das Vorticity-Feld type = 0: Transfer des in Feld[0] befindlichen Vektorfeldes in die Statistik-Felder Summe(x), Summe(y) -> Feld[1] Summe(x*x), Summe(y*y) -> Feld[2] Summe(x*y) -> [vorticity] IVAR[10] = n, Anzahl der eingegebenen Felder. type = 1: Mittelwert -> Feld[0] type = 2: Standardabweichung -> Feld[0] type = 3: Maximaler Fehler des arithmetischen Mittelwertes (Std.Dev./sqrt(N)) -> Feld[0] type = 4: Turbulenzgrad (Std.Dev./Average) -> Feld[0] Die Statistikfelder sind folgendermaßen definiert: - Vektorfeld[1]: Summe(x) und Summe(y) - Vektorfeld[2]: Summe(x*x) und Summe(y*y) - Vorticityfeld: Summe(x*y) ReturnValue 1, außer im Fehlerfall -1. Fehler können auftreten, wenn ein anderes als das Feld [0] selektiert ist, wenn weniger als 2 Felder in die Statistikfelder transferiert worden sind, wenn ein falscher type angegeben wird, oder wenn kein Vektorfeld definiert ist.
piv_plotfac	8, B/I	inter,xy_shift: integer; pfactor, sfactor: float; Einstellung von Größenfaktors zur PIV-Vektordarstellung, der Vektorfelddarstellung und dessen Verschiebung.. inter = 1: die Faktoren werden interaktiv eingefragt. Wenn jedoch einer der Werte (xy_shift, pfactor oder sfactor) auf -1 gesetzt ist, so wird dieser nicht eingefragt. inter = 0: die Werte werden über xy_shift, pfactor und sfactor eingegeben. Bei negativen Werten werden die internen Werte beibehalten. pfactor ist der Vektor-Scale-Faktor. Die Vektorlänge in Pixeln wird mit dem Faktor multipliziert und dann dargestellt. sfactor ist ein geometrischer Größenfaktor für das gesamte dargestellte Bild. Hiermit kann das Bild größer oder kleiner dargestellt werden. Die Vektorlänge ändert sich hierdurch nicht. Default ist der Wert 1.0. xy_shift ist ein zusammengesetzter Verschiebefaktor. Dabei ist in den oberen 16 Bit die y-Verschiebung angegeben, in den unteren 16-Bit die x-Verschiebung untergebracht. Beispiel: eine Verschiebung um y=100 und x=200 wird folgendermaßen eingegeben: y*65536 + x = 6553800 Diese Werte gelten für folgende Funktionen: piv_drawvect und piv_surface.
piv_freetable	0, B	kein Parameter; Freigeben der Vektor-Tabelle
data_params	3, B	Headers, DecPoint, Delimiter: integer; Parameter für die Darstellung von Daten beim Abspeichern auf Files. (siehe MACRO-Habndbuch)
color_bar	3, B/I	interactive, horz_vert, smallfont: integer; Kalibrierter Farbbalken für die Zuordnung von kalibrierten Werten zu Grauwerten. Wenn interactive==1, so hängt der Balken am Kursor und wird mit der linken Maustaste platziert. Ansonsten wird der Ort des einzelnen

Fadenkreuzes benutzt. horz_vert [0,1] bestimmt die Ausrichtung des Balkens. Mit smallfont=1 kann ein kleiner Font aktiviert werden, so daß eine größere Anzahl von Nachkommastellen dargestellt wird. In vertikaler Anordnung wird dann zusätzlich Exponentialformat benutzt.

color_bar_cal 8, B

autom, physunit: integer; min, max: float; Einstellung der Grenzen und der physikalischen Einheit des Kalibrierbalkens.
autom==0: Status der autom-Variable wird nicht geändert.
autom > 0: [1|2] Min/Max-Grenzen werden automatisch gesucht. Dies wird von bestimmten Funktionen der Zusatzmodule benutzt.
autom==-1: Interpretiere min als Offset, max als Gain-Faktor (noch nicht implementiert)
autom==-2: Set Min/Max (Bei der Funktion piv_drawvect werden Werte von min<0 zu min=0 gesetzt)
physunit: 0:[-], 1:[Länge=Unit], 2:[Geschwindigkeit=Unit/s],
3:[Beschleunigung=Unit/s2], 4:[Zirkulation=Unit2/s]
-1: [keine Änderung], -2: [interaktive Eingabe]
min, max: kleinster und größter Wert des Kalibrierbalkens, jeweils in der aktuellen (oder gleichzeitig gesetzten) physikalischen Einheit. Bei min=max=0.0 werden die Werte nicht geändert.
ReturnValue: IVAR[10]=autom, IVAR[11]=physunit,
FVAR[10]=min, FVAR[11]=max, FVAR[12]=goffs, FVAR[13]=gfact.
Jeder Bildspeicher hat eine eigene Colorbar-Einstellung, die bei Undo jeweils mitzurückgesetzt wird.

meas_control 4, B/I

inter, p1, p2, p3: integer; Setzen von Messungsparametern, ersetzt die alten Funktionen set_pol_gauss, set_aver_cross, meas_mono_rgb
inter = -2: Get Parameters:
IVAR[10] = CalUnit, IVAR[11] = CamCal
IVAR[12] = PolyWarp, IVAR[13] = RGB_Measure
IVAR[14] = PaletteView, IVAR[15] = Polynom_Gauss
IVAR[16] = Average_Cross
IVAR[17]=XCorrelation min size search, default=70 entspricht 0.7x
IVAR[18]=XCorrelation max size search, default=130 entspricht 1.3x
FVAR[10]=Pattern_Matching_Sigma min, default = 2.8
FVAR[11]=Pattern_Matching_Sigma max, default = 2.8
inter = -1: Dialogbox
inter = 0: Set Default Values
inter = 1: Automatische Kalibrierung: p1=cal_unit 0|1, p2=camcal 0|1, p3=polywarp 0|1
inter = 2: p1=RGB-Messung = 0|1, p1=-2: Gray-Calibrated Measure: kalibrierter Grauwert. Hierbei wird in der jeweiligen eingestellten Systemeinheit ausgegeben. Mit der Funktion "color_bar" kann kalibriert werden. Die Systemkalibrierung wird nicht benutzt, sie ist nur für die x- und y-Kalibrierung vorgesehen.
inter = 3: p1 = Palette Display in Status Line 0|1
inter = 4: p1=polynom_gaus 0|1: Setzen des Polynomannäherungs- oder des Gausskurvenannäherungs-Verfahrens für die subpixelgenaue Bestimmung von Peaks im 1D- oder 2D-Frequenzspektrum. 1D wird bei der Linien-Statistik benutzt, 2D bei der Partikel-Image-Velocimetry oder beim Projected Grid Verfahren (Zusatzmodule).
p2=aver_cross 0|1: Art der Subpixel-Maximum-Suche:
aver_cross = 0: es wird der grauwertgewichtete Flächenschwerpunkt der 3*3-Nachbarschafts-Fläche um das absolute Maximum berechnet.
aver_cross = 1: das subpixelgenaue Maximum wird über eine Kurvenanpassung bestimmt. Hierbei wird mit einem Polynom- oder Gaussansatz gearbeitet (siehe oben). Mit den lokalen Maxima über jeweils 3 Zeilen und 3 Spalten in der 3*3-Nachbarschaft um das absolute Maximum werden die Gleichungen zweier Kreise bestimmt, deren Schnittpunkt als Ort des absoluten Maximums genommen wird. Wenn sich nicht zwei Kreise bestimmen lassen, so werden Geradengleichungen benutzt. Diese Art der Subpixelbestimmung ist in der Regel wesentlich genauer.
inter = 5: Set Pattern_Matching_Sigma: p1 = 1000*Sigma.max, p2=1000*Sigma.min
inter = 6: Set XCorrelation min/max size search: p1 = min, p2=max
Value = -1: no change

2: Particle Tracking:

ptv_track	4, B (IM)	reference_buffer, subpix, passes, match_close: integer; Tracking-Funktion; Suchen der Partikel im ersten und zweiten PTV-Bild sowie Zuordnung aller gefundenen Partikel. Dabei wird das aktuelle Bild mit dem im reference_buffer liegenden Bild korreliert. Wenn reference_buffer==(-1), wird kein Referenzbild benutzt, sondern statt dessen die interne Zuordnungstabelle vom letzten Tracking bzw. Bildpaar. Auf diese Weise können komplette Bildsequenzen bei feststehender Partikelnummer-Zuordnung ausgewertet werden. subpixel= 0|1: Partikelortbestimmung über die Mittelpunktsberechnung des umschreibenden Rechtecks oder mit Subpixelgenauigkeit über eine Flächenschwerpunktberechnung. passes = 1..n: Anzahl der Iterationen bei der Partikelzuordnung. Ab dem 2. Pass wird eine Verbesserung der Zuordnung über eine 2-dimensionale Korrektur erzielt (über zwei bi-lineare Regressionen in x- und y-Richtung mit den sicher gefundenen Partikeln). match_close = 0|1: Wenn zu 1 gesetzt, wird versucht, sehr nahe aneinanderliegende Partikel, die sich um mehr als ihren gegenseitigen Abstand verschoben haben, über ihre Größe, den Abstand und den Winkel zueinander doch noch sicher zu identifizieren.
ptv_param	8, B	inter, iparam: integer; fparam1, fparam2: float: Eingabe von Parametern für die Partikel-Verfolgung. inter = 1: interaktive Dialogbox. iparam = 0: PTV Parameter Dialogbox iparam = 1: PTV Rejectioncriteria Dialogbox inter = 0: Eingabe einzelner Parameter: iparam = 1: in fparam1 und fparam2 werden die Variablen Maxdist1 und Maxdist2 erwartet. iparam = 2: in fparam1 und fparam2 werden die Variablen Maxdist3 und Maxdist4 erwartet. iparam = 3: fparam1=vmin, fparam2=vmax iparam = 4: fparam1=smin, fparam2=smax
ptv_show_list	1, I	list: integer; Anzeige der gefundenen Partikel der aktuellen Liste (list) durch Rechteckumrahmung und Partikelnummer. Bei list==-1 wird die Match-List angezeigt. Hierzu wird auch eine spezielle Look-Up-Table gesetzt. Einwandfrei zugeordnete Partikel werden mit Blauem Rechteck hervorgehobe, im zweiten Bild (im aktuellen) nicht mehr gefundene Partikel mit rotem Doppel-Kreis, und im zweiten Bild neu entdeckte Partikel mit grünem Doppel-Rechteck.
ptv_get_vectors	2, B	inter, n: integer; Interaktive Anzeige der PTV-Daten in einer Dialogbox oder Transfer Daten des angegebenen PTV-Vectors und der zugehörenden Partikelorte in die Macro-variablen: FVAR[10] = x1 FVAR[12] = x2 FVAR[14] = dx FVAR[11] = y1 FVAR[13] = y2 FVAR[15] = dy IVAR[10]=Match: 1=ok, 0=not found, -1=new particle
ptv_saveload	3, B/I	save_load, interactive, bits: integer; Speichern oder Laden der Verschiebevektoren im ASCII-Format auf den aktuellen geöffneten Daten-File oder auf einen eingefragten Daten-File. bits gibt die Art des Daten-Headers an und entspricht dem Parameter in der Funktion "piv-saveload". saveload = 0: Abspeichern der Vektordaten. Zuerst wird geprüft, ob ein Datenfile zum Schreiben im MACRO-Programm geöffnet ist. Wenn ja, so wird dieser benutzt. Wenn nein, wird geprüft, ob ein Systemdatenfile geöffnet ist. In diesem Falle wird dieser benutzt. Wenn kein Meßdatenfile geöffnet ist, so wird interaktiv ein Filename eingefragt und ein ASCII-File geöffnet und nach der Abspeicherung wieder geschlossen. Wenn vorher schon ein File geöffnet war, so wird dieser nicht geschlossen. Eine System-Kalibrierung wird berücksichtigt. Es wird dann allerdings immer in Meter [m] umgerechnet und gespeichert. Außerdem geschieht eine Umrechnung in das definierte System-Koordinatensystem. Die Daten werden im folgenden Format abgespeichert: *20* Kalibrierungs-Header System-Unit ist [pixel] oder [m] Cal_x, Cal_y, Origin_x, Origin_y, Rotation, Y_Down *22* ROI Coordinates Header x1, y1, x2, y2

24 Parameters
timestep, maxdist1, maxdist2, maxdist3, subpixel, passes, match_close
25 Listen-Länge-Header
Anzahl
26 Daten Header
1 dx dy x1 y1 x2 y2 match
2 dx dy x1 y1 x2 y2 match
... usw.

Das Abspeichern der Kalibrierung, der ROI-Koordinaten und der Parameter kann über die "bits"-Eingabe ein- oder aus-geschaltet werden. Wenn keine Daten vorhanden sind, wird folgender String abgespeichert: *27* No PTV data
saveload = 1: Laden einer Partikel-Liste. Wenn ein MACRO-Datenfile zum Lesen geöffnet ist, wird dieser benutzt. Wenn kein Datenfile geöffnet ist, so wird ein File interaktiv eingefragt. Eventuell vorhandene Listen werden gelöscht und neu definiert. Offene Files werden nicht geschlossen. Wenn jedoch interaktiv eingegeben wurde, so wird der File hinterher wieder geschlossen. Beim Einlesen werden alle Daten mit dem im File-Header gespeicherten Koordinatensystem wieder in die interne Pixel-Darstellung umgerechnet. Die Header-Einstellung (mit dem Parameter "bits") muß der Einstellung beim Abspeichern der Daten entsprechen. Wenn das Bit0 im Parameter "bits" gesetzt ist, so wird die zugehörige Kalibrierung miteingelesen. Entsprechendes gilt für die ROI-Koordinaten: entweder vorher auf die gleichen Werte einstellen oder beim Abspeichern der Daten mitspeichern (Bit2 von "bits" gesetzt).

Bit4	Bit3	Bit2	Bit1	Bit0
Params	Koeff.	ROI-Daten	alpha,beta,dist	cal_x_y

Beispiele: bits=21: speichere und lade Kalibrierung, ROI-Daten, PIV-Parameter. bits=5: speichere und lade Kalibrierung und die ROI-Daten. Bit1 und Bit3 werden nur vom FCM-Modul benutzt.
Wichtig: bits muß den gleichen Wert für Abspeichern und Einlesen der Daten haben, sonst können die Daten nicht wieder eingelesen werden! Beim Einlesen von Daten muß außerdem das Header-Flag im Measurement-Control-Menü die gleiche Einstellung haben wie beim Abspeichern.

ptv_draw_vectors	2, B (I)	type, color: integer; Anzeige der PTV-Verschiebevektoren. type: 1: End point 2: End cross 3: End circle 4: Connect line 5: Connect arrow color = -1 .. 0 .. 255 (-1 = interactive show by inverting)
ptv_gridipol	2, B	stepx, stepy: integer; Interpolation der verstreuten PTV-Vektoren über ein regelmäßiges Gitter. Die Daten liegen dann in PIV-Vectorfeldern.
ptv_dealloc_list	0, B	kein Parameter; Freigeben der PTV-Listen

Interferometrie (Manual Kapitel 8.10. und 14. und Zusatzmodul):

phase_shift	1, B	seq_3_4: integer; Phasen-Shift-Technik mit 3 oder 4 Bildern
phase_buffer	2, B	buf, val: integer; Setzen der Bildspeicher für die Phasen-Shift-Technik buf: Bildspeicher val: Nummer in der Phasen-Shift-Sequenz
unwrap	2, B	test, bias_buffer: integer; Phase-Unwrapping. Die Funktion sucht die günstigste Stelle in der ROI zum Starten, findet "Inconsistency"-Spots und tastet sich iterativ von allen Seiten an diese Spots heran. Inconsistency-Spots werden zu 255 gesetzt, alle 2PI-Phasensprünge werden durch Grauwertsprünge von 1 dargestellt, beginnend mit 0 am tiefsten Punkt des Bildes. Das aktuell angezeigte Bild muß das Sägezahnbild (sawtooth) enthalten, das Ergebnis wird in den bias_buffer geschrieben. test wird nicht benutzt.
compose	4, B	gain, biasbuffer, colorbar, corr: integer; combine bias-image and sawtooth-image to absolute elevation image. "gain" ist die Anzahl der kompletten Grauwertumfänge [0..255] zur Darstellung der Höheninformation, d. h. bei gain=1 werden alle Höhen im Interval [0..255] abgebildet, bei gain=2 werden die Höhenwerte im Interval [0..511] berechnet, das aufgrund der Grauwertbegrenzung des 8 Bit-Systemes dann im Intervall [0..255,0..255] abgebildet wird, usw. Eine HUE-Farbtabelle ist hier zur Darstellung sinnvoll. Zur Ausführung muß das sawtooth-image im aktiven Bildspeicher stehen, das Bias-image im "biasbuffer". Wenn colorbar = 1 ist, wird interaktiv ein Farbkalibrierbalken gezeichnet. In diesem Fall wirkt die Undo-Funktion "U" auf beides, d.h. das kombinierte Höhenbild und den Farbbalken. "corr" bestimmt die Art der geometrischen Umrechnung: corr = 0: ohne Änderung corr = 1: Geometrische Korrektur für Kamera und Projektor, ergibt Höhenbild aus Sicht des Projektors. corr = 2: Geometrische Korrektur nach Subtraktion eines Referenz-Sägezahnbildes, ergibt Höhenbild aus Sicht der Kamera. Die interaktive Meßfunktion für kalibrierte Grauwerte arbeitet nur bei gain=1, da sonst eine eindeutige Zuordnung eines kalibrierten Wertes zu einem Grauwert nicht möglich ist.
interfer_geom	8, B/I	interactive, alpha_beta_center: integer; angle, distance: float; Setzen der FCM- oder Interferenz-Parameter: Winkel und Abstand von Kamera und Projektor. alpha_beta_center = 0: angle, distance für Kamera eingeben alpha_beta_center = 1: angle, distance für Projektor eingeben alpha_beta_center = 2: distance ist Zentrum der optischen Achse der Kamera: 0: Reset to center of buffer 1: Center of last loaded image 2: Center or ROI 3: Manual Input in pixel on x-axis -90.0 < angle < 90.0: Winkel von Projektor und Kamera von der Flächennormale. Der Projektorabstand muß mit richtigem Vorzeichen eingegeben werden: Bei positivem Z-Wert ist der Abstand >0. Die Berechnung der Abweichung durch Kamera- und Projektorabstand wird nur durchgeführt, wenn dieser Abstand kleiner als unendlich eingegeben ist. Unendlich ist hierbei der Wert 1e30 (oder bei einem Projektoraufbau in negativer z-Richtung: -1e30). Wenn interactive=1, werden alle Werte interaktiv in einer Dialogbox eingegeben, die anderen Parameter der Macro-Funktion werden dann nicht benutzt. Im interaktiven Modus der Funktion kann auch ein künstliches Referenz-Streifenbild erzeugt werden. Dies wird im Macro mit der Funktion art_image (siehe unten) durchgeführt.
geom_saveload	2, I	save_load, interactive: integer; Laden oder Abspeichern der geometrischen Parameter für Kamera und Projektor. save_load = 0: Abspeichern save_load = 1: Laden der Parameter. Ein Filename wird interaktiv eingefragt. Die Extension ist immer und automatisch *.PRM. Es werden folgende Parameter gespeichert: Alpha (Camera) Beta (Projector) Camera Distance Projector Distance

		Fringe Width Fringe Phase ReturnValue ist 1 bei erfolgreicher Ausführung, 0 bei falscher (leerer) Filenameneingabe, und -1 bei Fehler (File not Found oder Schreib/Lesefehler)
art_image	8, B	sin_rect, dummy: integer; fringe_width, fringe_phase: float; Erzeuge künstliches Streifenreferenzbild. sin_rect: Es kann ein Rechteck-schwarz/weiß-Muster oder eine sinusförmige Grauwertverteilung der Streifen eingestellt werden. fringe_width ist die Breite eines Schwarz/Weiß-Streifens im angegebenen Projektorabstand (über interfer_geom eingegeben) bei einer Bildebene senkrecht zur optischen Achse des Projektors. fringe_phase ist die Phase von der Mitte der ROI aus gesehen.
ifer_export	4, B,	interactive, gain, biasbuffer, corrtype: integer: export ASCII image from sawtooth and bias image buffers. interactive = 0|1 gain: not used biasbuffer: image buffer with bias image contents - gain image buffer must be active! "corr" bestimmt die Art der geometrischen Umrechnung: corr = 0: ohne Änderung corr = 1: Geometrische Korrektur für Kamera und Projektor, ergibt Höhenbild aus Sicht des Projektors. corr = 2: Geometrische Korrektur nach Subtraktion eines Referenz-Sägezahnbildes, ergibt Höhenbild aus Sicht der Kamera. Die interaktive Meßfunktion für kalibrierte Grauwerte arbeitet nur bei gain=1, da sonst eine eindeutige Zuordnung eines kalibrierten Wertes zu einem Grauwert nicht möglich ist.

Fringe-Correlation-Method FCM (Projected Grid Method PGM)

(Manual Kapitel 8.11. und 14. und FCM-Zusatzmodul)

pgm_corr	3, B	passes, hamming, best_choice: integer; Korrelationsfunktion für die Bestimmung einer Verschiebung von aufprojizierten Lichtstreifen durch Bauteilverbiegung. Die Funktion arbeitet mit der Kreuzkorrelation über eine Linie. Daher sollte die Verschiebung der Streifen im wesentlichen 1-dimensional in horizontaler Richtung sein. Es wird in der aktuellen ROI gearbeitet. Winkel und Länge der Linie wird durch vorheriges Setzen mit der Funktion "pgm_size" durchgeführt. Die gewählte Linie sollte im wesentlichen senkrecht zu dem aufprojizierten Streifenmuster sein. Z. Zt. werden nur kleine Winkel zur Horizontalen unterstützt. Das Bild muß notfalls gedreht werden. Die Linienlänge wird intern auf einen Wert von (2**n) < (gewählte Länge) gesetzt, da die FFT mit Längen von (2**n) arbeitet. Außerdem müssen vorher mit der Funktion "pgm_param" die Linienbreite "width" und die Schritt- oder Gitterweiten in x- und y-Richtung "stepx" und "stepy" angegeben werden. Eine eventuell vorher mit der Funktion "pgm_autoshift" oder "pgm_shift" gewählte Basisverschiebung wird berücksichtigt und erlaubt es, auch größere als der halben Streifenmuster-periode entsprechenden Verschiebungen zu bestimmen. Es werden immer 3 Verschiebevektoren von abnehmender Wahrscheinlichkeit bestimmt und in die Felder [0], [1] und [2] gespeichert. Der wahrscheinlichste Vektor hat den Index "0", der zweitwahrscheinlichste den Index "1", und schließlich der Dritte den Index "2". Dieser Index kann bei den anderen Funktionen zur Darstellung oder Korrektur oder Umrechnung angegeben werden. Das zu vermessende Bild muß im aktuellen angezeigten Bildspeicher liegen, das Referenzbild muß über die Funktion "Reference Image" im PGM-Parameter-Menü definiert werden. Die Funktion liefert die Größe des Vektorfeldes (Anzahl in x,y-Richtung) in den Variablen IVAR[10]=x, IVAR[11]=y zurück. Wenn "hamming" zu 1 gesetzt wird, wird bei der Berechnung der FFT ein Hamming Window eingesetzt. Dies ist nur bei relativ langen Linien (> 4 Streifenperioden) sinnvoll, da sonst die Originaldaten zu stark verändert werden und es zu Fehlern kommen kann. Die Anwendung des Hamming-Windows hat jedoch den Vorteil, daß keine störenden sehr hohen Frequenzen mit hoher Amplitude durch Randprobleme auftreten, da ja die FFT immer als periodisch arbeitende Funktion angenommen werden muß. Wenn man also längere Linien mit der dann auftretenden Verschmierung durch Mittelung über die Linienlänge verwenden kann, so bringt das Hamming-Window eine höhere Genauigkeit. "passes" gibt die Anzahl von Iterationen an, um intern eine Basisverschiebung zu finden. Hierzu muß man verstehen, daß die Korrelationsfunktion natürlich wesentlich genauer arbeitet, wenn nur eine im Subpixelbereich liegende Verschiebung bestimmt werden muß. Bei einer großen Verschiebung zwischen Meßbild und Referenzbild sieht die Funktion im Fenster unterschiedlich aus, man erhält diffusere Korrelationsmaxima und eine Erkennung über die Korrelation ist daher nicht so genau. Daher ist es sinnvoll, die Basisverschiebung bis auf einen Pixel genau zur Verfügung zu haben. Da dies manuell nicht möglich ist, kann das System im zweiten Pass die ermittelte Verschiebung aus dem ersten Pass benutzen um das Korrelationsfenster direkt um die Basisverschiebung zu verschieben. Übrig bleibt dann nur noch die subpixelgenaue Bestimmung, die mit sehr hoher Genauigkeit durchgeführt werden kann, da das Korrelationsfenster bis auf den Bruchteil eines Pixels den gleichen Bereich des Streifenbildes abdeckt. Es macht keinen Sinn, mehr als 2 Passes auszuführen, da schon im ersten Paß bis auf Pixelgenauigkeit gerechnet wird. Mit der "best_choice"-Variable kann eingestellt werden, daß nach der Verschiebungsbestimmung die wahrscheinlichsten Vektoren in das Vektorfeld[0] übertragen werden. Dabei ist definiert, daß die wahrscheinlichste Verschiebung kleiner als die halbe Streifenperiode ist. Größere Verschiebungen können ja sowieso nur unzweideutig bestimmt werden, wenn ihre Basisverschiebung bekannt und über die pgm_autoshift-Funktion eingestellt wurde. Vor einem zweiten Durchlauf wird in jedem Fall diese Wahl getroffen, unabhängig vom "best_choice"-Parameter. Wenn der Active-Bildspeicher gleich dem Second-Source-Speicher ist, wird die Funktion nicht ausgeführt.
pgm_geom	8, B/I	interactive, alpha_beta_center: integer; angle, distance: float; Eingabe von FCM- oder Interferenz-Parametern: Winkel und Abstand von Kamera und Projektor, optische Achse der Kamera. Diese Funktion entspricht exakt der Funktion "interfer_geom" aus dem Interferometrie-Zusatzmodul, hat jedoch in der interaktiven Version eine andere Dialogbox-Überschrift. alpha_beta_center = 0: angle, distance für Kamera eingeben alpha_beta_center = 1: angle, distance für Projektor eingeben Hierbei ist der Wertebereich: (-90.0 < angle < 90.0), (-1.0E-30 < distance < 1.0E30)

"alpha" ist der Winkel der Kamera relativ zur lokalen Flächennormale [-90.0..90.0] in negativer mathematischer Richtung, default ist 0, d. h. die Kamera steht senkrecht zur aufgenommenen Fläche. "beta" ist der Winkel des Streifenprojektors [-90.0....90.0] in positiver mathematischer Richtung. Defaulteinstellung ist 45 Grad schräg zur Oberfläche. Mit diesen Defaulteinstellungen ist nach der Formel: d=-(cos(beta)/ sin(alpha+beta))*image_shift
die Höhenverschiebung gleich der Streifenverschiebung. Die Einstellung wird nur für den Display und beim Abspeichern und Einlesen der Daten benutzt. Intern sind alle Daten immer absolute Pixel-Verschiebungen im Bild.
"distance" ist der Abstand des Streifenprojektors oder der Kamera. Dieser Wert berücksichtigt, daß bei endlichem Streifenprojektorabstand die Projektionswinkel (oder bei der Kamera der Blickwinkel) am linken und rechten Bildrand geringfügig unterschiedlich sind. Diese Berechnung wird nur benutzt, wenn erstens das System in einer bestimmten Einheit kalibriert wurde und wenn zweitens der Abstand des Projektors oder der Kamera mit kleiner als +-1.0e30 in der jeweiligen Einheit (m, mm) eingegeben wurde. Der Projektorabstand muß dabei mit richtigem Vorzeichen eingegeben werden: Bei positivem z-Wert ist der Abstand >0. interactive = -1 setzt alle Werte auf die Defaulteinstellungen zurück.
interactive = 0: benutze angegebene Werte alpha_beta_center, angle, distance. angle ist dabei alpha oder beta, je nach Einstellung der Schaltervariable alpha_beta_center [0,1]. Bei alpha_beta_center=2 wird distance als Zentrum der optischen Achse der Kamera interpretiert. Hierbei sind folgende Werte möglich:
distance = 0.0: Reset zum Zentrum des Bildspeichers,
distance=1.0: Zentrum des letzten von Disk geladenen Bildes,
distance=2.0: Zentrum der aktuellen ROI,
distance=3.0: interaktiver Input des Zentrums in horizontalen Pixelkoordinaten.
interactive = 1: Interaktive Dialogbox zur Eingabe, die anderen Parameter der Macro-Funktion werden dann nicht benutzt. Im interaktiven Modus der Funktion kann auch ein künstliches Referenz-Streifenbild erzeugt werden. Dies wird im Macro mit der Funktion art_image durchgeführt.
interactive=2: Berechnung der FCM-Faktoren für die linke und rechte Seite der aktiven ROI und Ausgabe in einer Message-Box sowie Übergabe in den Variablen FVAR[10] = FCM-Factor_left, FVAR[11] = FCM-Factor_right.
ReturnValue ist immer 1.

geom_saveload 1, I

save_load: integer; Laden oder Abspeichern der geometrischen Parameter für Kamera und Projektor (siehe auch Interferometrie). Die drei Längenparameter werden immer in Pixel-Einheiten abgespeichert und intern gehalten. Für die Ausgabe und Benutzung werden sie in die aktuelle Einheit umgerechnet.
save_load = 0: Abspeichern
save_load = 1: Laden der Parameter.
Ein Filename wird interaktiv eingefragt. Die Extension ist immer und automatisch *.PRM. Es werden folgende Parameter gespeichert:

- Alpha (Camera)
- Beta (Projector)
- Camera Distance
- Projector Distance
- Fringe Width
- Fringe Phase

ReturnValue ist 1 bei erfolgreicher Ausführung, 0 bei falscher (leerer) Filenameneingabe, und -1 bei Fehler (File not Found oder Schreib/Lesefehler)

pgm_size 4, B

fx1,fy1,fx2,fy2: integer; Setzen von Länge und Ausrichtung des Korrelationsfensters. Setzen der Anfangs- und Endpunkte für die Linie, auf der die Streifenverschiebung bestimmt wird. Wenn die gewählte Linienlänge größer als (2**n), d. h. 16, 32, 64, 128, 256, usw. beträgt, wird sie von der Funktion pgm_corr verkürzt, wobei der Anfangspunkt im ersten (im interaktiven Modus im durchgezogenen Fadenkreuz) Punkt (fx1,fy1) bestehen bleibt. Hierbei wird die Linienlänge intern auf einen Wert von (2**n) < (gewählte Länge) gesetzt, da die FFT mit Längen von (2**n) arbeitet. Winkel zur Horizontalen sollten sehr klein sein. Wenn dies nicht der Fall ist, sollte man den Streifenprojektor vor der Bildaufnahme drehen oder später das Bild mit "rotate_angle" subpixelgenau interpoliert drehen. Achtung: für die Länge der Linie zählt nur Delta_x (fx2-fx1) oder Delta_y, und zwar der jeweils größere Wert.

pgm_param 4, B

width, stepx, stepy, reference: integer; Einstellung von Parametern zur FCM-Korrelation:

		width ist die Linienbreite, über die ein jeweiliger Grau-Mittelwert bestimmt wird [1..15], um Störungen und Rauschen zu vermindern. Sinnvoll sind Werte zwischen 3 und 7. stepx und stepy sind die Schrittweiten des rechteckigen Gitters, auf denen die Verschiebevektoren bestimmt werden [1..256]. reference ist die nummer des Bildspeichers in dem das Referenzbild abgelegt ist. Es wird immer die Verschiebung des aktuell angezeigten Bildes relativ zum Referenzbild berechnet.
pgm_param2	1, B/I	interactive: integer; Interaktive Eingabe aller FCM-Parameter in einer Dialogbox und Möglichkeit, zusätzliche Parameter anzugeben. interactive==1: Interaktiv werden z. Zt. sind folgende Parameter in einer Dialogbox eingegeben: line-length, line-width, line-angle, Schrittweite in x- und y-Richtung, der Referenz-Bildspeicher, die automatische Wahl des besten Vektors im Feld[0], 1 oder 2 Passes für die Korrelation, sowie die Kurvenanpassungsart bei der Peak-Bestimmung und ein Hamming-Window während der Fouriertransformation. interactive==0: Eingabe zusätzlicher Parameter, z. Zt. nicht unterstützt.
pgm_center	2, B/I	type, x: integer; Einstellung der Lage des Schnittpunktes der optischen Achse von Kamera und Projektor mit der Bildebene. type=0: Reset to center of ImageBuffer type=1: Center of last loaded image type=2: Center of actual ROI type=3: Interactive input in pixel from left hand ImageBuffer border type=4: use x in pixel (from left hand border) y-Coordinate always at image center.
set_pol_gauss	1, B	poly_gauss: integer; Die Korrelationspeaks der FCM-Methode werden mit Subpixel-Genauigkeit gefunden. Hierzu wird der höchste Peak im Korrelationsspektrum durch eine Polynom- oder Gausskurvenanpassung durch die den Peak umgebenden 3 Punkte genau bestimmt. Die Gaussanpassung empfiehlt sich besonders bei sehr dünnen Streifen (beim PIV-Verfahren für sehr kleine Partikel). Bei üblichen gleichmäßigen schwarz/weißen Streifen ist das Ergebnis für beide Methoden gleich. Z. Zt. ist dieser Schalter für die Funktionen von PIV, FCM, die Harmonischenbestimmung des Linienhistogrammes und die Maxima-Suche im FFT-Spektrum wirksam. Achtung: Funktion obsolete: bitte neue Funktion meas_control() nehmen!
pgm_autoshift	4, I	special_roi, small_large_mode, hamming, twopass: integer; Automatische Bestimmung der Basis-Verschiebung der Streifen, die über die halbe Streifenperiode hinausgeht, mit der Eingabe nur eines einzigen Referenzpunktes im Referenzbild und im Meßbild. Wenn special_roi=1 ist, so wird als erstes eine spezielle kleinere ROI eingefragt, die vollständig mit dem Streifenabbild bedeckt sein sollte. Wenn Teile der ROI keine Streifen aufweisen, so kann es passieren, daß die Suchfunktion falsche Werte liefert oder abbricht. Danach wird ein Referenzpunkt auf dem Referenzbild und dann auf dem Meßbild eingegeben, von dem aus das System versucht, die Basisverschiebung über die komplette ROI zu bestimmen. Die Suche nach oben und unten wird dabei mit einem Mustererkennungsalgorithmus durchgeführt, der auf einer 2-dimensionalen Kreuzkorrelation basiert. Die Größe der Korrelationsfenster richtet sich dabei nach der Länge der zuvor eingegebenen Fensters (Linienlänge) für die Verschiebungs-bestimmung, das etwa zwei Streifenperioden abdecken sollte. Mit dem Parameter "small_large_mode=0" kann bestimmt werden, daß das Korrelationsfenster nur die halbe Größe bekommt. Hiermit sollte nur im Falle auftretender Probleme experimentiert werden. Für die anschließende Suche in horizontaler Richtung nach links und rechts wird eine eindimensionale Korrelation mit doppelter Fensterbreite benutzt. Dies hat sich in Versuchen als Optimum erwiesen. In Fällen sehr großer Basisverschiebung kann es sein, daß diese Funktion keine vernünftigen Ergebnisse liefert. In diesem Fall muß die Basisverschiebung manuell über die Funktion pgm_shift eingegeben werden. Über den "hamming"-Parameter kann eingestellt werden, ob zur Liniensuche der Hamming-Filter während der Fouriertransformation benutzt werden soll. Dies ist defaultmäßig der Fall und sollte nur testweise abgeschaltet werden. Ohne Hamming-Filter kann es leicht zu einer Fehlbestimmung besonders bei kurzen Korrelationsfenstern kommen.
pgm_shift	8, B	get_store_interactive_x_y, const: integer; dx, dy: float; Setzen eines sogenannten "Basic Shift" (Basis-Verschiebung der Streifen über die halbe Streifenperiode hinaus). Mit der Korrelationsfunktion können wegen der periodisch wiederkehrenden Streifenmuster nur Verschiebungen bis zu einer halben Streifenperiode bestimmt werden. Eine stärkere Verschiebung wird als Verschiebung in die entgegengesetzte Richtung aufgefaßt. Dieses

Problem kann umgangen werden, indem das Bestimmungsfenster, d. h. die Bestimmungslinie in Richtung der Verschiebung mitbewegt wird. Wenn also bekannt ist, daß die Verschiebung etwa 10 Pixel beträgt, so kann dies dem System mitgeteilt werden. Die mögliche bestimmbare Verschiebung erhöht sich dann um diesen voreingestellten Wert, d. h. man kann dann Verschiebungen von 10 Pixeln plus der maximal halben Streifenperiode erkennen.. Wenn aufgrund von geometrischen Besonderheiten (z. B. Kamera und/oder Streifenprojektor nicht senkrecht zur aufgenommenen Fläche, Bewegung des Bauteils stark unterschiedlich auf der linken und rechten Bildseite) die Streifenverschiebung über die Bildbreite stark variiert, so kann dies über diese Funktion auch angegeben werden. Die Verschiebung in der gesamten ROI wird dabei über die Koeffizienten einer Ebenengleichung angegeben: BasicShift = x*gradient_x + y*gradient_y + const; Dabei zählen x und y von der linken oberen Ecke der ROI. Zum Setzen der Werte muß get_store zu 1 gesetzt werden. Wenn get_store==0, werden die aktuellen Werte in FVAR[10]=const, FVAR[11]=gradient_x und FVAR[12]=gradient_y geschrieben. Für den interaktiven Modus wird get_store_interactive_x_y zu 2 oder 3 gesetzt. Hier können dann die Verschiebewerte numerisch oder mit den Fadenkreuzen angegeben werden. Bei get_store_interactive_x_y = 2 werden die unterschiedlichen Werte im linken und rechten Teil der ROI eingegeben. Bei der numerischen Eingabe müssen die Basisverschiebungen für linken und rechten Rand der ROI eingegeben werden und ausserdem noch der y-Ort der Angabe (y=0 am oberen ROI-Rand - Diese y-Position wird nur verwendet, falls danach auch eine Variation der x-Verschiebung für oben und unten in der ROI eingegeben werden soll). Bei der Eingabe mit Fadenkreuzen kann dies irgendwo in der ROI geschehen, das System berechnet die Geradengleichung und ermittelt die Werte für die Ränder. Ebenso verfährt man mit den Verschiebewerten in x-Richtung am oberen und unteren Rand der ROI: sie werden bei get_store_interactive_x_y = 3 eingefragt, numerisch oder mit Fadenkreuzen. Bei numerischer Eingabe bitte wieder die Basisverschiebung genau am oberen und unteren ROI-Rand eingeben, und zwar etwa in der horizontalen Mitte der ROI. Bei Eingabe mit Fadenkreuzen wieder irgendwo im der ROI, das System kümmert sich um den Rest. Wenn eine unterschiedliche Verschiebung oben und unten eingegeben werden soll, so muß zuerst eine Verschiebung für links und rechts eingegeben worden sein. Während der Eingabe mit Fadenkreuzen kann mit den Ziffertasten zwischen den beiden Bildern (Active und Reference) umgeschaltet werden. Das erste Fadenkreuz (durchgezogen) wird dabei auf eine Markierung im Referenzbild gesetzt, das zweite Fadenkreuz (gestrichelt) auf die gleiche Markierung im Meßbild.
Rückgabe: FVAR[10] = Basis-Verschiebung links oben
FVAR[11] = Gradient in x, FVAR[12] = Gradient in y

pgm_cal 8, B

type, calmaskbuf: integer; shift, nr: float; Kalibrierung der Verschiebedaten: Berechnung der Koeffizienten einer Kalibrierebene zur Nachbearbeitung der Vektorfelder mit Hilfe eines Pixelmaskenbildes: für linear 2-dimensional verzerrte Vektorfelder.
type = 1: Berechnen der Koeffizienten der Kalibrierebene, mit der später bei der Anzeige oder beim Abspeichern von Vektordaten diese durch Multiplikation korrigiert werden können. Hierzu muß ein Maskenbild vorliegen, innerhalb dessen Vektorverschiebungen als sinnvoll, d. h. mit bekannter Verschiebung, angesehen werden. "calmaskbuf" ist der Bildspeicher, in dem das Maskenbild vorliegt. Dieses kann z. B. erzeugt werden, indem man mit einem 2D-Vektorzug die interessierenden Bereiche umfährt und dann den Vektorzug in ein leeres Bild mit dem Grauwert 255 plottet und mit dem ImagePaint-Programm ausfüllt.
"shift" ist hierbei die eingestellte exakt bekannte vertikale Verschiebung der Oberfläche oder z. B. die Dicke einer Folie, die zum Kalibrieren auf die zu vermessende Fläche aufgetragen wurde. Diese Dicke wird in Pixeln oder bei Kalibrierung in der jeweiligen Einheit (Meter, Millimeter) angegeben. Zur Berechnung der Koeffizienten der Kalibrierebene werden die Referenzvektordaten aus dem Vektorfeld[nr] benutzt, die Korrekturfaktoren werden intern abgelegt. Sie können ausgelesen oder mit einem Vektorfeld [0|1|2] abgespeichert werden. Beim Hereinladen eines Vektorfeldes werden die Koeffizienten dann wieder geladen. Versuchsaufbauwerte alpha, beta, Kamera- und Projektor-Distance werden bei der Kalibrierung berücksichtigt, d. h. man kann auch ein bereits durch diese Angaben justiertes Ergebnis nachkalibrieren. Dies kann übrigens auch zur Kontrolle der geometrischen Versuchsparameter benutzt werden: bei richtiger Angabe sollte die Kalibrierebene Werte sehr nahe an 1.0 aufweisen und keine Gradienten haben. Die Korrektur läuft nach folgender Gleichung ab: Korrigierte Verschiebung = f(x,y) = Verschiebung/=Ordinate + x*Gradient_x + y*Gradient_y.
type = 2: Reset der Koeffizienten der Kalibrierebene: Gradient in x- und y-Richtung=0.0, Ordinate=1.0.
type = 3: Anzeige der Koeffizienten in einer Messagebox.

Eingabe von "maskbuf" und "shift" nur bei type=1. Die Koeffizienten werden immer (außer bei Fehler) in den Macro-Variablen ausgegeben:
FVAR[10] = Ordinate (linke obere Ecke der ROI)
FVAR[11] = Gradient x
FVAR[12] = Gradient y
ReturnValue = -1 bei Fehler: zu wenig Speicher, zu kleines Vektorfeld usw.

pgm_getvect 4, B/I

interactive,x,y,nr: integer; Abfragen der Vektordaten (x,y) von Feld[nr] aus dem Macro-Programm. Die Daten werden folgendermaßen übergeben:
FVAR[10] = x[Pixel] relativ zur oberen linken Ecke der ROI
FVAR[11] = y[Pixel] relativ zur oberen linken Ecke der ROI
FVAR[12] = dx[Pixel] oder kalibriert [Unit]
FVAR[13] = dy[Pixel] oder kalibriert [Unit]
Die x- und y-Werte werden bei Systemkalibrierung auch kalibriert ausgegeben. In der interaktiven Version wird eine Datentabelle als Dialogbox in folgender Form angezeigt: nx, ny, x, y, dx, dy, best_choice. Diese Funktion arbeitet genau wie piv_getvect, aber mit Berücksichtigung von Streifenprojektor und Kamerawinkel, d. h. multipliziert mit -cos(beta)/sin(alpha+beta). Hierbei wird auch der Projektorabstand berücksichtigt, der in unterschiedlichen Winkeln "alpha" und "beta" am linken und rechten Bildrand resultiert, wenn das System kalibriert wurde und der Projektorabstand oder Kameraabstand kleiner als 1.0e30 der jeweiligen Einheit ist (m, mm)

pgm_function 8, B

type, nr: integer; maskbuf, kernel: float; Filterung oder Korrektur des FCM-Vektorfeldes mit Möglichkeit, über den "maskbuf" beliebige Arbeitsregionen zu setzen:
type==1: Filtern des Vektorfeldes[nr] mit einer Gauss-Maske. Wenn hierbei die Variable "param=maskbuf" ungleich Null ist, so wird der Bildspeicher |(int)maskbuf| als Maske bei der Filterung verwendet. Vektoren, die außerhalb der Maske liegen, werden nicht berücksichtigt. Wenn maskbuf<0 ist, werden die an die Maske gerade angrenzenden Vektoren noch mitberücksichtigt (nur für Spezialeffekte).

kernel=+-1.0: Vertikaler 3*1-Kernel (1,2,1)
kernel=+-2.0: Horizontaler 1*3-Kernel (1,2,1)
kernel=+-3.0: Vertikaler 3*3-Kernel (1,2,1,2,4,2,1,2,1)
Wenn kernel<0.0 ist, sind die Vektoren an den Rändern der ROI fest und werden nicht geändert.

type==2: Bestimmung des wahrscheinlichsten Verschiebevektors (best choice) aus den von der Funktion pgm_corr gegebenen 3 Vektoren. Diese Funktion arbeitet nach folgendem Schema: Benutze Vektor[0], wenn die Verschiebung kleiner als die halbe Streifenperiode ist, sonst teste Vektor[1] und Vektor[2]. Wenn alle Verschiebevektoren zu groß sind, so wird dennoch Vektor[0] benutzt. (nr und param werden nicht benutzt)
type==3: Bestimmung des wahrscheinlichsten Vektors anhand der Anordnung der Nachbarvektoren. Diese Funktion ist noch nicht implementiert.
type==4: Löschen aller Vektoren außerhalb der weißen Maske im Bildspeicher, der durch maskbuf=(integer)param angegeben ist. Wenn maskbuf = 0 ist, wird die Funktion nicht ausgeführt.
type==5: Mittelung aller Null-Vektoren mit den Nachbarn mit wie oben angegebenem Kernel (kernel=1.0|2.0|3.0). Null-Nachbarn werden dabei nicht berücksichtigt. Null-Vektoren entstehen z. B. durch Zurückweisungskriterien während der Displacementbestimmung oder durch Löschen mit der "Delete w/Mask"-Funktion (type 4). Diese Funktion kann damit zur Extrapolation von Vektoren außerhalb des Masken-Bildes benutzt werden. Dies ist ein wichtiger Schritt zur sinnvollen Darstellung maskierter Bereiche.
type = 6: Arithmetik mit zwei Vektorfeldern:
kernel = 1: Add: field[n] := field[n] + field[param]
kernel = 2: Sub: field[n] := field[n] - field[param]

pgm_surface 2, B

nr, x_y_xy: integer; Anzeige der Verschiebevektoren (oder für PIV Modul auch desVorticity-Feldes) als interpoliertes Grauwertbild.
nr ist der gewünschte Vektor der drei von pgm_corr bestimmten Verschiebe-Vektoren, normalerweise immer der wahrscheinlichste, d.h. "0"
x_y_xy: Berechnung der geplotteten Verschiebungsfläche aus x-Verschiebungwerten, y-Werten, oder dem Betrag der Werte:
x_y_xy = 0: Benutze nur x-Verschiebung
x_y_xy = 1: Benutze nur y-Verschiebung
x_y_xy = 2: Benutze sqrt(x*x+y*y) für die Verschiebe-Fläche
Bei der Fringe Correlation Method werden nur x-Verschiebungen bestimmt, alle anderen Darstellungen sind hier sinnlos, sie werden im PIV-Modul benutzt. Nach der Darstellung

ist es sinnvoll, einen kalibrierten Farbbalken mit der Funktion color_bar anzuzeigen. Die Darstellung erfolgt normalerweise automatisch mit komplettem Grauwertebereich für den Bereich der vorkommenden Verschiebewerte. Dies kann jedoch in bestimmten Fällen, z. B. bei großen Fehlervektoren unsinnig sein. Eine geringe Differenz von Verschiebungen wäre dann im Grauwert nicht mehr sichtbar gegenüber der falschen starken Verschiebung des Fehlvektors. Es kann auch sein daß man bestimmte Werte als Max/Min-Werte festeinstellen möchte. Hierzu dient die Funktion color_bar_cal, deren Parameter hier erläutert werden sollen:
autom = 1 automatische Wahl der Parameter zur vollen Ausnutzung des Grauwertebereiches, setzen der internen Variablen als Offset und Gain
autom = 2 automatische Wahl der Parameter zur vollen Ausnutzung des Grauwertebereiches, setzen der internen Variablen als Min und Max (Defaulteinstellung)
autom = -1 benutze min/max als offset und gain zur Bestimmung der Grauwertverteilung
autom = -2 benutze min und max zur Bestimmung der Grauwertverteilung.
Die Funktion color_bar_cal muß vor der Funktion pgm_surface aufgerufen werden, da die aktuellen min/max-Werte hier benötigt werden. In die Macrovariable werden folgende Werte abgelegt:
FVAR[10] = offset FVAR[11] = gain
FVAR[12] = min FVAR[13] = max
sfactor der Funktion piv_plotfac ist für diese Funktion gültig. Man kann damit das Bild größer oder kleiner darstellen.

color_bar 3, B/I

interactive, horz_vert, smallfont: integer; Kalibrierter Farbbalken für die Zuordnung von kalibrierten Werten zu Grauwerten. Wenn interactive==1, so hängt der Balken am Kursor und wird mit der linken Maustaste platziert. Ansonsten wird der Ort des einzelnen Fadenkreuzes benutzt. horz_vert [0,1] bestimmt die Ausrichtung des Balkens. Mit smallfont=1 kann ein kleiner Font aktiviert werden, so daß eine größere Anzahl von Nachkommastellen dargestellt wird. In vertikaler Anordnung wird dann zusätzlich Exponentialformat benutzt.

color_bar_cal 8, B

autom, physunit: integer; min, max: float; Einstellung der Grenzen und der physikalischen Einheit des Kalibrierbalkens.
autom==0: Status der autom-Variable wird nicht geändert.
autom > 0: [1|2] Min/Max-Grenzen werden automatisch gesucht. Dies wird von bestimmten Funktionen der Zusatzmodule benutzt.
autom==-1: Interpretiere min als Offset, max als Gain-Faktor (noch nicht implementiert)
autom==-2: Set Min/Max (Bei der Funktion piv_drawvect werden Werte von min<0 zu min=0 gesetzt)
physunit: 0:[-], 1:[Länge=Unit], 2:[Geschwindigkeit=Unit/s],
3:[Beschleunigung=Unit/s2], 4:[Zirkulation=Unit2/s]
-1: [keine Änderung], -2: [interaktive Eingabe]
min, max: kleinster und größter Wert des Kalibrierbalkens, jeweils in der aktuellen (oder gleichzeitig gesetzten) physikalischen Einheit. Bei min=max=0.0 werden die Werte nicht geändert.
ReturnValue: IVAR[10]=autom, IVAR[11]=physunit,
FVAR[10]=min, FVAR[11]=max, FVAR[12]=goffs, FVAR[13]=gfact.
Jeder Bildspeicher hat eine eigene Colorbar-Einstellung, die bei Undo jeweils mitzurückgesetzt wird.

pgm_drawvect 4, B/I

col, line_arrow, nr,edit: integer; Interaktives Editieren, Anzeigen und Plotten der PIV oder FCM-Vektoren.
col: [-2,-1,0..255]: Grauwert zum Plotten. Wenn col=-1, wird nur invertiert und nach weiterem Mausklick wieder invertiert. Bei col=-2 wird der Vektorlänge ein Grauwert zugeordnet. Je nach Stellung der "autom"-Variable der Funktion color_bar_cal wird dies automatisch über den gesamten Grauwertebereich [0..255] durchgeführt oder nur für einen bestimmten Ausschnitt.
line_arrow (Achtung: [1|2]) bestimmt, ob Vektorpfeile oder nur Geradenstücke gezeichnet werden sollen. Geradenstücke sind z. B. sinnvoll bei einer Autokorrelation, bei der die Richtung der Vektoren nicht bekannt ist. Die Länge ist die tatsächliche Verschiebung in Pixeln multipliziert mit dem pfactor der Funktion piv_plotfac.
Wenn line_arrow negativ [-1|-2] angegeben wird, so wird die Vektorlänge konstant ausgegeben, nämlich mit der Länge pfactor der Funktion piv_plotfac in Pixellängen. Dies macht nur Sinn, wenn zugleich die Vektorlänge grauwertkalibriert wird mit col==-2.
nr ist der gewünschte Vektor der drei von der Korrelations-routine bestimmten wahrscheinlichsten Vektoren.
edit muß normalerweise auf "0" gesetzt sein. Wenn edit=1, wird in den interaktiven Edit-

Modus verzweigt: col wird intern dann auf -1 gesetzt und die Vektoren werden durch Invertieren dargestellt. Ein Mausklick in der Nähe eines Vektors bewirkt, daß zyklisch der jeweils nächstwahrscheinliche Vektor dargestellt wird, immer in der Reihenfolge 0,1,2,0,1,2... Als erstes bei Funktionsaufruf wird immer der mit "nr" angegebene Vektor dargestellt. Nach Beendigung des Editiervorganges mit der rechten Maustaste oder der ESC-Taste fragt das System, ob die eingestellten Änderungen gespeichert werden sollen. Bei Zustimmung werden die aktuell dargestellten Vektoren mit dem Index "nr" gespeichert, die anderen Vektoren mit den übrigen Indices werden zyklisch verschoben. (Die Funktion arbeitet wie piv_drawvect, nur daß der Streifenprojektor-winkel beta und Kamerawinkel alpha berücksichtigt wird) Im Edit-Modus kann zusätzlich ein Vektor mittels einem Gauss-Kernel in einer Konvolution seinen Nachbarn angepaßt werden. Dazu wird der gewünschte Vektor angeklickt und dabei zusätzlich die Shift-Taste oder die mittlere Maustaste gedrückt. Jedes Ausführen dieser Funktion bewirkt eine neue Konvolution und gleicht damit den Vektor besser an die Nachbarn an. Diese Änderung kann nicht mehr rückgängig gemacht werden. Im Zweifelsfalle also das Vektorfeld vorher abspeichern.

piv_plotfac 8, B/I

interactive, dummy: integer; pfactor, sfactor: float; Mit den Plotfaktoren wird die dargestellte Länge der PIV/FCM-Vektoren und die absolute Größe des dargestellten Vektorfeldes eingestellt. Bei pfaktor=1.0 ist die Länge der Vektoren gleich der jeweiligen Verschiebung in Pixeln. Bei kleinen Verschiebungen empfiehlt es sich, den pfactor zu vergrößern. sfactor sollte normalerweise immer auf 1.0 stehen. Bei sfactor!=1.0 wird das ganze Bild verkleinert oder vergrößert. Bei Werten < 0.0 werden die intern eingestellten Werte nicht verändert. Wenn also nur ein Wert geändert werden soll, so muß der andere Wert mit z. B. -1.0 eingegeben werden. Wenn interactive=1, werden die Werte eingefragt, sofern sie nicht als <0.0 angegeben sind.

pgm_saveload 4, B/I

saveload, nr, interactive, bits: integer; Abspeichern oder Her-einladen der Vektordaten des Vektorfeldes mit der Feldnum-mer nr. Die Funktion arbeitet wie piv_saveload, allerdings wird der Kamera- und Streifenprojektorwinkel berücksichtigt.
saveload = 0: Abspeichern der Vektordaten. Zuerst wird geprüft, ob ein Datenfile zum Schreiben im MACRO-Programm geöffnet ist. Wenn ja, so wird dieser benutzt. Wenn nein, wird geprüft, ob ein Systemdatenfile geöffnet ist. In diesem Falle wird dieser benutzt. Wenn kein Meßdatenfile geöffnet ist, so wird interaktiv ein Filename eingefragt und ein ASCII-File geöffnet und nach der Abspeicherung wieder geschlossen. Wenn vorher schon ein File geöffnet war, so wird dieser nicht geschlossen. Es wird Feld "nr" abgespeichert. Hierbei wird eine metrische Kalibrierung und der Projektor- und Kamerawinkel und eventuell auch der Projektorabstand mit berücksichtigt. Außerdem geschieht eine Umrechnung in das definierte System-Koordinatensystem. Die Daten werden im folgenden Format abgespeichert:
20 Kalibrierungs-Header System-Unit ist [pixel] oder [m]
Cal_x, Cal_y, Origin_x, Origin_y, Rotation, Y_Down
21 Experimental Setup Header
alpha, cam.-dist, beta, proj.-distance
22 ROI Coordinates Header
x1, y1, x2, y2
23 Koeffizienten: Kalibrierebene, Basic Shift, 3D-Correction
cal_ord gradx grady shift_ord gradx grady 3d_ord gradx grady
24 Geometry Parameters
dx, dy, angle, stpx, stpy, 2pass, pol/gauss, hamm, best, opt.axis
25 x,y-Feldgröße-Header
Anzahl_x Anzahl_y
26 Daten Header
x1 y1 dx1 dy1
x2 y2 dx2 dy2
... usw.
Das Abspeichern der Kalibrierung, der experimentellen Daten, der ROI-Koordinaten, der Kalibrierebenen und der geometri-schen Parameter kann über die "bits"-Eingabe ein- oder aus-geschaltet werden. Wenn keine Daten vorhanden sind, wird folgender String abgespeichert: *27* No PIV/FCM data
saveload = 1: Laden eines Vektorfeldes in Feldnummer nr. Wenn interactive==0, so wird ein vorhandener MACRO-Datenfile, der zum Lesen geöffnet war, benutzt. Wenn kein Datenfile geöffnet ist, so wird die Funktion beendet und 0 zurückgegeben. In der Regel wird beim Einlesen das gesamte Vektordatenfeld gelöscht und neu definiert, da sich ja andere Feldgrenzen ergeben könnten. Falls dies nicht erwünscht ist und in ein vorhandenen Feld geladen werden soll, so kann dies mit interactive==-1 getan werden.

Wenn dabei das Vektorfeld nicht mit dem schon definierten übereinstimmt, so wird das neue Feld Vektor für Vektor in das vorhandene Feld eingelesen und dabei eventuell abgeschnitten. Offene Files werden nicht geschlossen. Bei interactive==1 wird ein Filename eingefragt und der File hinterher wieder geschlossen. Beim Einlesen werden alle Daten mit dem im File-Header gespeicherten Koordinatensystem wieder in die interne Pixel-Darstellung umgerechnet. Basic Shift und eine eventuelle 3D-Positionskorrektur werden beim Einlesen der Daten jedoch nicht wieder berücksichtigt, da dies für die Daten selbst irrelevant ist und nur interessant ist, um die Entstehung der Daten nachzuvollziehen. Die Header-Einstellung (mit dem Parameter "bits") muß der Einstellung beim Abspeichern der Daten entsprechen. Wenn das Bit0 im Parameter "bits" gesetzt ist, so wird die zugehörige Kalibrierung miteingelesen. Entsprechendes gilt für die Versuchsaufbau-Parameter alpha, beta und den Projektor-Abstand. Entweder vorher auf die gleichen Werte einstellen oder beim Abspeichern der Daten mitspeichern (Bit1 von "bits" gesetzt). Der Projektorabstand wird nur berücksichtigt, wenn metrisch kalibriert wurde. Ein Abstandswert in Pixeln wird nicht unterstützt. Für richtiges Funktionieren der Abstandskorrektur muß auch die ROI-Größe dem System bekannt sein. Vor dem Laden also für die gleiche ROI-Einstellung sorgen oder die ROI mit abspeichern (Bit2 in "bits" gesetzt).

Bit4	Bit3	Bit2	Bit1	Bit0
Params	Koeff.	ROI-Daten	alpha,beta,dist	cal_x_y

Beispiele: bits=31: speichere und lade cal_x_y, alpha, beta, distances, ROI-Daten, Koeffizienten der Kalibrierebene und alle FCM-Parameter. bits=5: speichere und lade cal_x_y und die ROI-Daten

Wichtig: bits muß den gleichen Wert für Abspeichern und Einlesen der Daten haben, sonst können die Daten nicht wieder eingelesen werden! Beim Einlesen von Daten muß das Header-Flag im Measurement-Control-Menü die gleiche Einstellung haben wie beim Abspeichern.

pgm_statistic 1, B

type: integer; Statistische Funktionen mit mehreren Vektorfeldern.
type = -1: Reset: Clear Field[1], Field[2], Vorticity-Field
type = 0: Add: Field[0]->Statistic Fields:
Sum(x), Sum(y) -> Field[1]
Sum(x*x), Sum(y*y) -> Field[2]
Sum(x*y) -> Vorticity Field
type = 1: Average
type = 2: Standard Deviation
type = 3: Standard Deviation / sqrt(n)
type = 4: Turbulence (Std.Dev. / Average)
ReturnValue = -1, wenn nicht Field[0] aktiv ist, <2 Felder aufsummiert wurden, oder ein unzulässiger type angegeben wurde.

piv_freetable 0, B

kein Parameter; Freigeben des für die Vektortabelle von PIV/FCM-Daten benötigten Speicherbereiches. Die Routine wird intern automatisch bei Aufruf der pgm_corr oder der entsprechenden PIV-Routinen piv_cross und piv_auto aufgerufen.

art_image 8, B

sin_rect, dummy: integer; fringe_angle, fringe_phase: float: Erzeuge künstliches Streifenreferenzbild (siehe auch Interferometrie). Hierzu müssen alle Parameter in der PGM-Geometry-Dialogbox eingestellt sein, wie Projektorabstand und -winkel und Kameraabstand und -winkel. Zusätzlich wird hier noch der Divergenzwinkel und die Phase der projizierten Streifen benötigt.
sin_rect = 0 erzeugt eine sinusförmige Streifengrauwert-verteilung,
sin_rect = 1 erzeugt schwarz/weiß-abwechselnde Streifen.
dummy wird z. Zt. nicht verwendet.
fringe_angle ist der Divergenzwinkel eines Schwarz/Weiß-Streifens in Radianten.
fringe_phase ist die Phase von der Mitte der ROI aus gesehen.

pgm_angle 4, B/I

interactive, vect, mskbuf, color: integer; Bestimmung der Torsion oder des Anstellwinkels einer Fläche gegenüber der y-Richtung, d. h. also um die Bildschirmhorizontale. Hierbei wird durch alle in y-Richtung liegenden Vektoren eine Regressionsgerade gebildet. Der Winkel der Regressionsgerade wird in einer Dialogbox oder direkt graphisch auf dem Bildschirm ausgegeben und kann auf Wunsch abgespeichert werden. Hierbei werden Winkel für alle x-Positionen bestimmt. Wenn ein von Null verschiedener Masken-Bildspeicher (mskbuf) angegeben wird, so wird dieser als Maske benutzt. Nur Vektoren, an denen die Maske einen von Null verschiedenen Wert aufweist, werden für die Regression berücksichtigt. Bei nichtinteraktiver Ausführung wird entweder auf den Bildspeicher gezeichnet oder es werden die Werte lediglich

abgespeichert. Dabei wird auf einen vorhandenen Systemdatenfile geschrieben. Wenn keiner geöffnet ist, wird ein Filename eingefragt. Dabei wird nach folgendem Schema gespeichert:
**** x, Blade Angle of Attack
x1 theta1
x2 theta2
...
interactive = 0: save data noninteractively
interactive = 1: open dialogbox
interactive = 2: plot (draw) angles on image buffer using color [-1,0..255]. color = -1 inverts graphics.
Der ReturnValue kann im Fehlerfall zu -1 gesetzt werden, z. B. wenn kein Speicher für die Winkel-Liste gefunden wird, oder wenn keine FCM-Berechnung durchgeführt worden war.

pos3d_setup 2, B/ID

type, param: integer; Einstellung aller Parameter und Berechnung der Transformationsmatrizen zur 3D-Positionsrekonstruktion. Diese Funktionen sind im Realtime/3D-Positions-Zusatzmodul beschrieben. Die 3D-Rekonstruktion kann dazu benutzt werden, eine eventuelle Verschiebung eines Objektes, dessen Oberfläche bestimmt werden soll, als Korrektur während der FCM-Berechnung zu berücksichtigen. So kann z. B. bei der Rotorblatt-Oberflächenanalyse eines Hubschraubers sich der komplette Hubschrauber bewegt haben und damit die Analyse der Blattoberfläche ohne Korrektur sinnlos werden. Zur Korrektur werden in der Regel zwei zusätzliche Kameras benutzt, die die Lage des Objektes ständig verfolgen. Hieraus wird über eine 3D-Rekonstruktion eine Lageverschiebung oder Drehung bestimmt, die dann automatisch zur Korrektur der FCM-Daten benutzt werden kann. Eine genauere Beschreibung der Funktion findet man im Kapitel 3D-Rekonstruktion. Hier werden nur die für die FCM wichtigen Parameter dargestellt:
type = 3: Eingabe der Parameter für eine Korrektur von Oberflächenverbiegungsdaten (FCM-Modul) in einer Dialogbox.
type = 4: Art der Korrektur bei der FCM-Oberflächenanalyse:
param=0: keine Korrektur der FCM-Daten.
param=1: benutze dz, pitch und roll für die FCM-Korrektur
param=2: benutze nur die vertikale Translation dz für die FCM-Korrektur
param=3: Interaktiver manueller Input der Korrektur-Ebenengleichung mit Ordinate an der linken oberen Ecke der ROI und den Gradienten in x- und y-Richtung.

Positionsbestimmung 2D/3D (Manual Kapitel 8.12. und 14. und Zusatzmodul)

rt_position	4, B / I	type, param1, param2, param3: integer; Echtzeit-Positionsbestimmung und Marker- oder Objektverfolgung **type = 2:** Real Time Object Tracking, Echtzeitverfolgung der Marker. Die ROI´s werden dabei immer auf den neuen gefundenen Mittelpunkt der Marker gesetzt. Bei einer Split-Screen-Einstellung können ROI´s nicht die Trennungslinie zum anderen Kamerabild-ausschnitt überschreiten. param1 und param2 sind die Kameras, von denen eingelesen wird [0..3]. Bei negativer Kameranummer wird nach Kameraselektion auch der Synchronisationseingang auf diesen Kameraeingang geschaltet, falls die Hardware dies unterstützt. Bei param3=1 werden auch die 3D-Koordinaten der Raumpunkte bestimmt, wenn die 3D-Transformationsmatrizen gesetzt sind. Wenn dieser Parameter nicht dem Schalter in der "Marker Parameter" Dialogbox entspricht, wird eine Warnung ausgegeben. Für Split-Screen Tracking wird die Scanline oder Column, bei der auf die zweite Kamera umgeschaltet wird, in der rt_param2-Funktion gesetzt. Rückgabewerte siehe unten. **type = 3:** Markerverfolgng wie in (type=2), jedoch nicht in Echtzeit, sondern nur mit dem/den aktuell angezeigten Bildspeicher/n. Hiermit können - über ein Macro-Programm - ganze Sequenzen ausgewertet werden, indem in einem Loop die einzelnen Sequenz-bilder auf den Bildschirm gebracht werden. param1 und param2 werden nicht benutzt. Bei param3=1 werden auch die 3D-Koordinaten der Raumpunkte bestimmt, wenn die Transformationsmatrizen gesetzt sind. Rückgabewerte für type==2 (lediglich IVAR[10] und IVAR[11]sowie STRING[1] gelten auch für type==3): IVAR[10] = Markeranzahl IVAR[11] = Schlimmster Fehler (Error Code) eines Markers IVAR[12] = Anzahl der Bestimmungen, d.h. der Trackingdurchläufe IVAR[13] = Trigger Condition (z. Zt. nur DDE-HotKey-Character) IVAR[14] = Ringbuffer Max Distance. IVAR[15] = Ringbuffer Skip_Crash (crash (-1) or skipped (>0)) IVAR[16] = Ringbuffer Delay IVAR[17] = EndCondition; IVAR[18] = Endtrigchar: Ent-Trigger-Character, falls DDE benutzt wurde IVAR[19] = FramesLost: Fehler im Frame Grabber? FVAR[10] = TimeStampMax [in 0.1ms] STRING[1] den ErrorCode als ASCII-String, z. B. ‘‘000100’’: 6 Marker, der 4te Marker liegt zu nah am Rand. Zusätzlich wird der RingBufferDelay, der I/O-Status, der FrameIndex, der TimeCode für zwei Kameras und ein eventueller Frame-Lost-Fehler ebenfalls für zwei Kameras ausgegeben. Statt dieser zusammengefassten Frame-Lost-Werte lassen sich hier je 5 spezielle Fehlermeldungen für zwei Kameras ausgeben, nämlich "FrameLost", "BandwidthError", "ExternalTriggerIgnored", "IncompleteData", "SkippedImages". Dies wird mit der Funktion rt_param2(10,-1, write|0x20,-1) geschaltet, wo write die unten beschriebene Schreibkonfiguration ist, meist 0x05, 0x06, oder 0x07. RingBufferDelay ist dabei die Anzahl der Bilder, um die der Auswertezeiger hinter dem Bildeinlesezeiger zurückgefallen ist. Dieser Wert kann mit der mittleren TimeCode-Differenz zweier Bilder zu einer Delay-Zeit umgerechnet werden, um die dann ein Echtzeitwert zu spät geliefert wurde. I/O-Status ist eine zuvor definierte externe digitale Schnittstelle, die 1 oder 0 sein kann. Der FrameIndex ist normalerweise die Bildnummer im Sequenzspeicher, kann jedoch für bestimmte Zwecke auf 0 zurückgesetzt werden. Die Art des TimeCodes wird in der AcqConfigurationBox ausgewählt. Folgende Error Codes sind definiert: 0: no error, 1: leaving ROI or screen soon (minimal distance to ROI border has to be 1/4 marker diameter, otherwise ROI will not be moved anymore) 2: marker was reconstructed from nearest neighbourhood, use with care! 3: marker rejected by some rejection criteria 4: marker touches border of search ROI 5: no marker in ROI, nothing touches ROI 6: too many valid markers in ROI 7: 3D-Warning: Yaw/Roll unsave 8: 3D-Error, Matrix Inversion Error (3 points on a line or similar problem), or File Error **type = 0, 1, 4, ..., n:** Alle anderen type-Werte verzweigen in die unten beschriebene rt_param2-Funktion.

rt_params	0, ID	no parameter; Interaktive Dialogbox zum Setzen aller Parameter für die Objektverfolgung. Hier können auch weitere Unter-Dialogboxen geöffnet werden, in denen zusätzliche Parameter eingestellt werden, z.B. Camera-Calibration Paramter, Real-Time-Parameter, Objekt-Rejection-Kriterien, usw...
rt_param2	4, B / I	type, param1, param2, param3: integer;

Set all parameters for Real-Time-Position-Measurement and marker or objekt tracking and for other real-time functions. If a parameter can be switched on or off or has positive values only, i.e. for array indices, using -1 will not change that parameter. This can also be useful for setting only one or two of the 3 possible parameters, or for requesting a parameter value without changing it.
Setzen aller Parameter für Echtzeit-Positions-messung und auch für andere Echtzeit Funktionen. Wenn ein Parameter mit 0 oder 1 aus- oder eingeschaltet werden kann oder wenn nur positive Werte zugelassen snd, z.B. bei Array-Indices, dann bewirkt die Eingabe von -1, das der Parameter nicht geändert wird. param1 ist in einigen Fällen eine Freigabe für interaktive Fehlermeldungen.

type = 0: Define marker positions, param1 = marker dia, param2 = max shift, param3 = marker count
type = 1: Check roi's (animation): param1 = interactive messages, param3 = time [ms]
type = 2: Real time tracking, use rt_position(2,...)
type = 3: Off-line single tracking, use rt_position(3,...)
type = 4: Get/Set xy-data and other data of marker (timecode...)
type = 5: Transfer Mrk to Ptim or reconstruct 3D->2D, Ptob->Ptim and transfer back to Mrk
type = 6: Adjust roi size for adaptation during measurements, param2 = new max shift
type = 7: Save marker roi's: param1 = interactive messages , param2 = marker set (0|1)
type = 8: Recall marker roi's: param1 = interactive messages , param2 = marker set (0|1)
type = 9: Show Marker, Threshold, and Extrapolation Params
type =10: Set graphics, savedata, calc3d_set_camera_axes:
type =11: Set filter, individual size validation, size min/max parameter
type =12: Basic Marker Parameter: marker diameter, marker count, max shift
type =13: Inhibit-Isotop, View, and Auto Adaptation Params
type =14: Adapt Threshold: interactive messages, adapt threshold and also area, adapt single marker
type =15: Preset single marker position and try to find the marker
type =16: Enlarge ROI and Ring Buffer Savety Params
type =17: Marker adaptation parameters
type =18: Split/Two-Screen Setting:
type =19: Edit Marker location
type =20: Set NumberSamples: tracking will run NumberSamples samples, default=-1 for ever;
type =21: Preset Marker location and keep other parameters as area, threshold, gray
type =22: Preset Marker[param1] Area=param2, Gray-Level=param3
type =23: Preset Marker[param1] Thr=param2 and TBox=param3

type = 0: Define marker position interactively: Initialisierung der Objektverfolgung und interaktive Eingabe der zu verfolgenden Objekte durch einfaches Anklicken mit der Maus.

param1: marker diameter, in doubt better select it a bit larger - ungefähre Größe der Objekte/Marker in Pixeldurchmessern, im Zweifel eher größer angeben.

param2: maximum shift of markers between consecutive images - maximal zugelassene Verschiebung der Marker von einem zum nächsten Bild in Pixelgrößen. Dieser Wert steuert die Größe der Such-ROI.

param3: marker count in all images (stereo) (for 2D: param3=-1: don't know) - Gesamtanzahl der Marker in allen Teilbildern, die definiert werden sollen. Das Programm bricht automatisch nach Definition der vorgegebenen Anzahl von Markern ab und zeigt dann die Such-ROI´s kurz nacheinander zur Überprüfung an. Wenn param3 zu (-1) gesetzt wird (nur für 2D-Auswertungen), ist die Anzahl beliebig und man muß die Definition mit der rechten Maustaste abbrechen (intern ist jedoch die Anzahl begrenzt, z. Zt. auf 100 Marker).
Undo der letzten definierten Markerposition ist möglich mit der rechten Maustaste oder ESC, während die Markerinformation in der Statuszeile angezeigt wird.
Über den Parameter 1 (Bit4) der Funktion rt_param2(9,...) kann eingestellt werden, ob die Marker per Mausklick oder mit Gummiband-Vektoren definiert werden.
type = 1: Check search ROIs (Animation) - Anzeige der ROI´s, innerhalb derer die Objekte gesucht werden. Je nach Einstellung des "show_n_lr"-Parameters (siehe rt_param2(9,...) werden die Marker-ROIs einfach in Grau dargestellt und von 0 bis n-1 durchnumeriert oder in Signal-Rot dargestellt und stereometrisch richtig von 1..n/2 für das linke und rechte Bild getrennt durchnumeriert.

prm1: interactive messages; prm2 not used

prm3: display time [ms] - Zeige Marker nacheinander als Animation, Geschwindigkeit kann interaktiv geändert werden: 0..t[ms]. prm3 = -1: Anzeige aller Marker gleichzeitig mit Nummer in beiden Bildern.
type = 2: nicht definiert! (Real time tracking, use rt_position(2,...))
type = 3: nicht definiert! (Off-line single tracking, use rt_position(3,...))

type = 4: Get/Set xy-data and other data of marker - Auslesen der letzten gefundenen Markerpositionen oder der Bildzeitstempel oder Eingabe von subpixelgenauen Markerpositionen für spätere 3D-Rekonstruktion sowie Eingabe eines Zeitstempels. prm1 kann -1 zum Auslesen von Parametern sein oder auch [0..Markernummer) zur Eingabe von genauen Markerpositionen sowie -2 zur Eingabe von Zeitstempeln.
Get data - Ausgabe:
prm1 = -1: return Mk[].cg to FVAR[], time stamps to IVAR[] - Markerpositionen/Zeitstempel auslesen
prm2 = [0..D3D_depth): get timestamps 1st->IVAR[13], 2nd->IVAR[14] - der Zeitstempel des jeweiligen Bildes wird in IVAR[13] ausgegeben. In IVAR[14] wird dann der Zeitstempel der zweiten Kamera ausgegeben, falls Stereo-Bildübernahme eingeschaltet ist. prm2 = -1: keine Rückgabe der TimeStamp Werte.
prm3 = [0..Marker): get return values for marker, IVAR[10] = Rt.count, IVAR[11] = Pos_Errorworst, FVAR[10]/[11]=corrected xcg/ycg, FVAR[12]/[13]=not corrected pixel based xcg/ycg - Rückgabe verschiedener Markerparameter: in IVAR[10] steht dann die Anzahl der Marker, in IVAR[11] der maximale ErrorCode und in FVAR[10]/[11] die x,y-Komponente des Markers (subpixelgenauer Flächenschwerpunkt incl. Origin und Calibration). Die Werte sind (je nach Einstellung) kalibrierte Werte mit dem Nullpunkt im definierten Ursprung. In FVAR[12]/[13] stehen die nicht korrigierten Werte. prm3 = -1: no timestamp return values - keine Rückgabe der Daten.
Set data - Eingabe:
prm1 = -2: preset time stamp (prm3) to seq.buffer[prm2] of first (left) camera - setze Zeitstempel der ersten (linken) Kamera, dabei sind: prm2=Bildindex, prm3=Zeitstempel.
prm1 = [0..Marker): preset xcg, ycg - Eingabe der Markerpositionen. Dabei sind:
prm2 = Xcg * 1000.0 (Mk[prm1].xcg = prm2/1000.0)
prm3 = Ycg * 1000.0 (Mk[prm1].xcg = prm3/1000.0)
Preset xcg, ycg (uncorrected) and then preset Search-ROI and mark marker as new and never found - Es werden dann Marker-Xcg, Ycg, sowie die Koordinaten der Such-ROI gesetzt. Alle anderen Parameter der Marker werden so indiziert, daß das System weiß, daß sie noch nicht gefunden worden sind. Nach der Eingabe des letzten Markers wird automatisch das interne Marker-Definitionsflag gesetzt, so daß eine Suche oder auch Rekonstruktionen vorgenommen werden können. Parameter wie Marker-Anzahl, Size, Shift und 3D-Origin müssen zuvor gesetzt sein.
Beispiel für nachträgliche Auswertung: Loop über rt_param2(4,0..n,x1000,y1000), dann pos3d_calc(n/2,0,0).
type = 5: Transfer Marker to Ptim or reconstruct 3D->2D, Ptob->Ptim and transfer back to Marker - Transfer Marker Position to internal 3D-Position measurement variables or back to Marker-Variables.
prm1 = transfer direction = 0|1:
0: transfer Marker -> internal 2D-Ptim variable (prm3 = overall 3D marker count (nm2))
1: Ptob->Ptim reconstruction, then transfer internal 2D-Ptim variable -> Marker and Search-ROI and try to find the Marker in the Stereo-images.
prm2: point set to use for transfer: 0=reference, 1=result points
prm3: overall 3D marker count, just for verification (Pos3d.count/2 | Rt.count/2=nm2) - Anzahl der Raumpunkte [1..n], d. h. halbe Anzahl der Marker Diese Anzahl muß genau halb so groß sein wie die Anzahl der Markerpositionen in den beiden Teilbildern der beiden Kameras (wird intern verifiziert!).
Beispiel: Vorgehensweise zum Rücktransfer in 2D-Koordinaten zur automatischen Eingabe von Markerpunkten ohne manuelle Definition durch Anklicken: Lade sinnvolle 2D-Parameter, lade 3D-Positionsmatrix, lade 3D-Meßobjekt, Aufruf der Funktion rt_param2(5,1,0,n), teste 2D-Markerpositionen. Über Rt.markersize kann man in gewissen Grenzen bestimmen, ob ein neuer Marker gefunden wird, auch wenn er nicht exakt in der Mitte der Such-ROI liegt. Es wird bis Rt.markersize/4 vom Mittelpunkt aus gesucht. Return Values:
IVAR[10] = nm2, d. h. Anzahl der Raumpunkte, IVAR[11] = Missing Marker Count, d. h. Anzahl der nicht gefundenen Marker. Im Fehlerfall in IVAR[10] Anzahl aller Marker
* type = 6:
* param1 =, param2 = new max shift
type = 6: Adjust roi size for adaptation during measurements - Neue Einstellung für die maximale Verschiebung der Marker zwischen zwei aufeinanderfolgenden Bildern. Diese Funktion kann auch nach einer Markerdefinition noch durchgeführt werden, wenn z. B. festgestellt wird, daß sich die Marker schneller bewegen als vorausgesehen.
param1 = 0|1: interactive error message - Bei param1 = 1 werden eventuell Meldungen ausgegeben
param2 = new max shift value in pixel. Alle Such-ROIs werden geändert, so daß mit dem neuen Shift gesucht wird.
type = 7: Save marker ROIs - Abspeichern der Marker-Suchbereiche (ROIs) auf interne Variablenfelder. Es können zwei Sets von Marker-ROIs gespeichert werden.
param1 = 0|1: interactive messages - Bei param1 = 1 werden eventuell Meldungen ausgegeben
param2 = 0|1: Speichern auf ROI Set 0 oder 1
type = 8: Recall marker ROIs - Wiederherstellen der Marker-Suchbereiche (ROIs) von den internen Variablenfeldern. Es können zwei Sets von Marker-ROIs restauriert werden.
param1 = 0|1: interactive messages - Bei param1 = 1 werden eventuell Meldungen ausgegeben
param2 = 0|1: Laden von ROI Set 0 oder 1
type = 9: Show Marker, Threshold and Extrapolation Params - Graphikanzeige, Thresholdbestimmung,

Extrapolation verlorener Marker bei den nachfolgenden Markerbestimmungen.
prm1 = show_n_lr = type of marker ROI display:
Bit0 = 0: show gray ROIs with marker numbers from 0..n-1;
1: show signal red ROIs with marker numbers 1..n/2 for stereo buffers left/right
Bit 1: x-position 0|1 = right|left - Ziffernanzeige in x:right=0 / left=1
Bit 2: y-position 0|1 = down|up - Ziffernanzeige in y: below=0 / above=1
Bit 3: distance from center 0|1 = small|large - Ziffernanzeige: distant=0 / close=1
Bit 4: 0:use mouse click on marker for definition,
1: use rubber vector lines for search
prm2 = use global threshold 1..255 (set Obj.threshold), prm2 = 0: use individual local threshold
prm3 = extrapolate lost markers 0|1|2. 0: Feature off; 1: use average from last 5 marker locations;
Achtung: When Extrapolation of Markers is set to "1", the last 5 displacements are averaged. This is used to preshift the search ROI. If markers are not found, ROI is moved by this averaged shift to follow non-visible markers relatively savely. Not found markers are assumed on the shifted ROI-center. This method can be used for single tracking as well: the average values are not cleared at function start, if extrapolation is set to "1". Handling: 1st tracking set extrapolation to "0", before 2nd tracking set to "1" to keep average displacement. Keep "1" for following single trackings. Use with care!
2: use nearest neighbors for reconstruction; lost marker will be reconstructed from the nearest neighboring markers. Very save and accurate method as long as object is rigid.
Returns: IVAR[10]=show_n_lr; [11]=useglobalthr; [12]=extrapol; IVAR[13]=Obj.threshold
type = 10: Graphics, Save Data on File, Calculate 3D Position and set camera axis:
prm1 = show graphics:
Bit0 = 0|1 = show circles of approx marker size,
Bit1 = 0|1 = show search-ROIs
Bit16 = 0x10000 = 1: use colors for status code:
green=ok(0), white=border(1), yellow=reconstructed(2), magenta=rejected(3),
red=errorcode(4,5,6), blue=new_never_found (-1)
prm2: Save Data to File:
bit0: save 3D data,
bit1: save 2D data,
bit2: write enable (1=on by default),
bit3: use CRLF for 2D data after each x,y-coordinate pair.
bit4: 0 = ASCII, 1 = Binary, saves some space on file
bit5: 1: save extensive error/status information with 16 values instead of only 8 values
param3 = 0|1|2|3: Rt.rt3dpos 0|1 (setting to 0 clears 3dpos flag and camera axis, setting to 1 sets 3dpos flag but does not set any camera axis. 2 sets camera axis to ROI center, 3 sets camera axis to image buffer center). - 3D Positionsbestimmung. Bei 0 lösche 3dpos-Flag und interne Kameraachse, bei 1 setze 3dpos-Flag ohne Änderung der Kameraachse, bei 2 setze Kameraachse in die Mitte der ROI, bei 3 setze Kameraachse in die Mitte des Bildspeichers.
Achtung: Benutzung des write enable Bits: nur wenn dieses Bit gesetzt ist, werden tatsächlich Daten weggeschrieben. Dies dient dazu, bei Echtzeit-Hochgeschwindigkeitsauswertungen ohne zeitaufwendiges Öffnen eines Files steuern zu können, ab wann Daten geschrieben werden sollen. Die Steuerung geschieht dann z.B. über die DDE-Schnittstelle durch Senden eines speziellen Hot-Keys (0x8a=on, 0x8b=off).
Wenn Headerzeilen eingeschaltet sind, werden die Datem mit folgenden Headern auf den File geschrieben:
35 Pos Tracking Start: Time hh:mm:ss (oder TimeStamp Typ)
36 x1 y1 x2 y2 x3 y3 .... StatusCode Delay IOStatus Index TimeCode LF
x1[1] y1[1] x2[1] y2[1] x3[1] y3[1]... 0000... 0 0 n nnnnn 0
31 Point x y z WorstError (3D-Position)
1 x[1] y[1] z[1] 0
2 x[2] y[2] z[2] 0
.....
37 Pos Tracking Stop: Time hh:mm:ss (oder TimeStamp)
39 Enable Tracking At "time" (when enable write was toggled)
Returns: IVAR[10]=graphics; [11]=savedata; [12]=calc3d; IVAR[13]=originx, IVAR[14]=originy
type = 11: Filter for search-ROI, individual size validation - Filterfunktionen in der jeweiligen Marker-ROI, außerdem Individual Size Validation und hierfür die Min/Max-Size-Eingabe:
prm1 = filter: Bit0: de-interlace all ROIs 0|1
Bit1: invert all ROIs 0|1
Bit2: Laplace3x3sharpen all ROIs 0|1
Bit3: Laplace5x5sharpen all ROIs 0|1
Bit4: Gauss3x3 all ROIs 0|1
Bit5: Gauss5x5 all ROIs 0|1
Bit6: FlatFieldCorrection FFC in all ROIs 0|1 (für Stereo unterschiedliche Buffer 0, 1)
Bit7: Contrast Enhancement, not yet implemented
Gauss Filter wird zuerst benutzt, falls Laplace und Gauss beide eingeschaltet sind. Der größere Kernel (5x5)

wird zuerst angewandt, falls beide Kernel eingeschaltet sind.
prm2 = 0|1: individual size validation: off|on. Von..bis in % der ursprünglichen Größe während der Definition des jeweiligen Markers: maxmin in der 16bit-Belegung:
prm3 = (max<<16) | min, beides 16Bit-Werte: individual Marker Size.
Returns: IVAR[10]=filter; [11]=sizevalidation; [12]=maxmin;
type = 12: Marker Size, Marker Count, and Max-Shift:
prm1 = 1..n: Set Marker Size (pixel): (New Marker Definition necessary)
prm2 = 1..m: Set Marker Count: (New Marker Definition necessary)
prm3 = 1..x: Set Maximum Marker Shift for new definitions (from one image to next). Hierbei wird der Shift nur für eine neue Marker-Definition gesetzt. Bei vorhandener Definition rt_param2(6,...) benutzen!
Returns: IVAR[10]=markersize; [11]=markercount; [12]=maxshift;
IVAR[13]=oldshift, IVAR[14]=Marker_Defined
type = 13: Inhibit-Isotop, Display Live Image, and Automatic Adaption to Illumination:
prm1 = 1: sets Inhibit-Isotop parameter. Now, if more than one marker are found at the same location, only the first one is accepted, all others are rejected
prm2 = -3|0|1: view image by paintscreen (-3: do not show status line), prm2 = -1 or -2: don't change
prm3 = 0|1: Automatic adaption to new illumination during tracking: uses average of histogram of square of twice the markers size.
Returns: IVAR[10]=dummy; [11]=paintscreen; [12]=autoadapt;
type = 14: Adjust/Adapt threshold to actual illumination - Adaption der Grauwertschwelle zur Markererkennung an neue Lichtverhältnisse. Hierbei wird mit den bekannten Markerorten eine neue Objektbestimmung durchgeführt, bei der der Objektdurchmesser und die optimale Grauwertschwelle iterativ neu bestimmt wird. (evtl. mehrfach aufrufen!)
prm1 = 0|1: interactive messages - Bei param1=1 werden eventuell Meldungen ausgegeben
prm2 = 0|1: 0: Adapt Gray Level, 1: Adapt Gray Level And Area
prm3 = -1 for all markers, or 0..n for single marker adaptation
type = 15: Preset and single marker position and try to find the marker - Eingabe von neuen Markern mit ihren pixelgenauen Orten. Die Software versucht dann auch, diese Marker zu finden:
prm1 = n, d. h. die aktuelle Markernummer
prm2 = xm,
prm3 = ym
If not found, the marker is marked as new and never found.
type = 16: Enlarge ROI and Ring Buffer Savety Params - Automatische Vergrößerung der Such-ROI, falls Marker sie berührt
prm1 = 0..n: Allow enlarging ROI by 0..n pixel if marker not found - Anzahl der Pixel, um die die Such-ROI vergrößert wird. Bei prm1==0 ist die Funktion abgeschaltet.
prm2 = 0|1|2: fall_back_skip: no fallback | fall back ok | skip allowed (Zurückfallen im Ringspeicher) Additionally here is coded the minimum processing delay <<16 - Zusätzlich wird im oberen 16-Bit-Wort noch ein MinimumDelay gesetzt: MinimumDelay<<16. Default ist 1. Ein Delay von 0 ist einstellbar, dann sollten jedoch bei Stereo-Acquisition beide Kameras perfekt aufeinander synchronisiert sein.
prm3 = 2..(D3D_depth-2): Ring Savety: used when fallback==1, stops acquisition when savety margin is reached, to prevent crash into ring buffer - Bildübernahme und Auswertung im Ringspeicher stoppt, bevor der Auswertezeiger minus RingSavety vom Bildeinlesezeiger überschritten wird.
Returns: IVAR[10]=enlargeroi; [11]=fallback; [12]=ringsavety;
type = 17: Marker adaptation parameters: - Besetzen von Variablen zur automatischen Markersuche während des Trackings - nur zum Test...
prm1 = 0|1|2|3|4|5|6:
0: thrs min bright
1: thrs max dark
2: thrs / gray difference in %
3: in 2D-Box (defined size / actual area) difference in %
4: thrs percentage of average of all markers
5: gray percentage of average of all markers
6: area percentage of average of all markers
prm2: Input value - Einzugebender Wert
prm3: not used
Returns: IVAR[10]=minbright; [11]=maxdark; [12]=graydifference;
IVAR[13]=areadifference, IVAR[14]=thrs_percentage
IVAR[15]=gray_percentage, IVAR[16]=area_percentage
type = 18: Split/Two-Screen Setting - Setzen der Split-Screen oder Two-Screen-Parameter:
prm2 >= 0: Column/Line for splitscreen configuration
prm1=splitscreen: 0: not split screen, 1: horizontal besides each other, 2: vertically one above the other.
prm2 = -1: prm1 = twoscreen = 0|1 = off/on
prm3: not used
Returns: IVAR[10]=splitscreen; [11]=twoscreen; [12]=split_h_v;
IVAR[13]=split_column, IVAR[14]=split_line

type = 19: Edit marker interactively (set correct stereo buffers in stereo mode!)
(for stereo, use markers in active image):
prm1 = setnew_keep = 0|1. 0: set edited markers as new and never found.
1: keep old gray|thrs/area of the edited markers
prm2 = direction: -1:transfer vector list->marker; 0: edit; 1: transfer marker->vector list
prm3 = interactive: ask for permission to delete old vector list
type =20: Set NumberSamples: tracking will run NumberSamples samples, default=-1 for ever;
<-1: no acquisition, speed test with actual images, in regular 2D image buffer!
type =21: Preset Marker location and keep other parameters as area, threshold, gray
param1 = [0,1,2...Rt.count): preset xcg, ycg: Mk[].xcg = param2, Mk[].xcg = param3, and then preset Search-ROI, but keep all other parameters. Caution: integer values to pixel accuracy only.
type =22: Preset Marker[param1] Area=param2 and Gray-Level=param3; Return values in IVAR[10], [11]
param2, param3 = -1: do not change value, but still return value
type =23: Preset Marker[param1] Thr=param2 and TBox=param3; Return values in IVAR[10], [11]
param2, param3 = -1: do not change value, but still return value

rt_saveload	2, B/I	save_load, interactive: integer; Laden oder Speichern der Marker-Parameter für eine Markersuche. Wenn interactive==0, wird auf den oder vom zum Schreiben oder Lesen geöffneten und aktivierten Datenfile geschrieben oder davon gelesen. Achtung: ältere .TPR-Files sind nicht in allen Fällen kompatibel zu neueren Versionen des picCOLOR-Programmes, da neue Parameter hinzugekommen sind. Diese Parameter sind z. B.: - Anzahl der Marker, - Markergröße, - Bildkonfiguration, d. h. Einzelbild für 2D-Bestimmung, oder Split-Screen oder 2-Bild-Konfiguration für 3D-Bestimmung, - Ursprung des Bildes, d. h. Ort der optischen Achse der Kamera, - Koordinatenursprung im Bild (Origin_x, Origin_y) - Maximale Markerverschiebung zwischen zwei Bildern, - Benutzung einer Global-Threshold - Marker Extrapolation verschiedener Art - Verschiedene Filter - Abspeicherung der 2D-Markerorte auf File, - Berechnung der 3D-Rekonstruktion. Wenn zum Zeitpunkt des Abspeicherns schon Marker definiert sind, so werden alle ihre zusätzlichen Daten (Such-ROI's, Schwellwerte, Pixelflächen, zwischengespeicherte Such-ROI's) ebenfalls abgespeichert. Es wird ein Binärfile abgespeichert mit der Extension .TPR (für Tracking Parameter).

Echtzeit-Funktionen (Manual Kapitel 8.12. und 14. und Zusatzmodul)

Allgemeine Funktionen zur Echtzeitsteuerung:

get_trigger 2, B/I

check_waitstart_waitstop, interactive: integer: Abfragen des Auftretens einer Triggerbedingung oder Warten bis zum Auftreten einer Triggerbedingung.
check=0: Funktion kommt sofort zurück und gibt an, ob eine Triggerbedingung vorliegt
check_waitstart_waitstop=1|2: Funktion wartet solange, bis eine Triggerbedingung eintritt. Bei eventuell gesetztem Start/Stop-Delay wird bei waitstart_waitstop=1 der Start-Delay genommen, bei waitstart_waitstop=2 der Stop-Delay. Erst nach der angegebenen Wartezeit kommt die Funktion zurück. Diese beiden Fälle schalten die Bildübernahme ein, falls sie abgeschaltet ist. Hierzu muß die Bildübernahme jedoch initialisiert sein (setup_acquisition)
interactive = -1|0|1: Ein- oder Ausschalten einer interaktiven Eingabemöglichkeit zum Triggern. Bei interactive=0 ist die interaktive Eingabe abgeschaltet. Bei interaktive=-1 kann lediglich mit der rechten Maustaste oder der ESC-Taste eine Messung abgebrochen werden. Bei interactive=1 kann eine Triggerbedingung mit der linken Maustaste oder der RETURN-Taste eingegeben werden oder die Abbruchbedingung gegeben werden.
In einer Triggersequenz mit Start und Stop muß der erste Aufruf mit der Variable check_waitstart_stop=1 gegeben werden. Nur dann wird der interne Timer gesetzt, der einen 2. Trigger innerhalb einer minimalen Wartezeit verhindert. Der High Resolution Timer im picCOLOR-System muß eingeschaltet und verwendbar sein.
Der Ergebniswert der Triggerung wird in der Variable IVAR[0] zurückgegeben. Zur Benutzung müssen also mit "stop_on_error" die Macro-Warnungen oder der Abbruch abgeschaltet sein. IVAR[10] gibt weitere Triggerbedingungen zurück, z. Zt. nur den DDE-Character, der zur DDE-Triggerung benutzt wurde.
Return Value ist 1 bei eingetretener Triggerbedingung, -1 bei User-Abbruch, 0 bei nicht eingetretener Triggerbedingung und check = 0. Bei DDE-Triggerung wird der empfangene Charakter in IVAR[10] zurückgegeben

set_trigger 3, B/I

type, prm1, prm2: integer; Setzen von Triggerbedingungen:
type = -2: interaktive Dialogbox
type = -1: Interaktive Anzeige der Soft-Trigger-Linie im Bild, auf der nach Grauwertänderungen gesucht wird. Außerdem Übergabe aller Trigger-Parameter in die Return-Variablen. Wenn dies im Batch-Modus benutzt werden soll, muß zuvor msg_interactive 0 gesetzt werden.

IVAR[10] = bitwise: bit0|bit1|bit2|bit3: Triggerart: Soft | External | DDE | Comm
IVAR[11] = Trigger-Line Horz/Vert IVAR[12] = Trigger Line Length
IVAR[13] = Trigger Line x-Start IVAR[14] = Trigger Line y-Start
IVAR[15] = Trigger Line Threshold IVAR[16] = Trigger Line Low/High
IVAR[17] = Trigger Line Minimum Delay between start and stop
FVAR[10] = Measure Time FVAR[11] = Measure Time at Trigger
FVAR[12] = Delay Start FVAR[13] = Delay Stop
FVAR[14] = Hard I/O FVAR[15] = ASCII Character
FVAR[16] = Check Trigger Line FVAR[17] = Check I/O
FVAR[18] = Sync Trigger Line FVAR[19] = Sync I/O

type = 0: Abschalten aller Triggerbedingungen (Interaktive Triggerung ist aktiviert)
type = 1: Setze Triggerbedingung mit prm1: wenn prm1=-1 ist, dann setzt
prm2: bitwise: bit0|bit1|bit2|bit3: Triggerart: Soft | External | DDE | Comm
Different trigger methods can be enabled concurrently. DDE-Hot-Key is always active during real time tracking for future extensions!
prm1 = 0: disable prm1 = 1: Software-Trigger
prm1 = 2: Hardware-Trigger prm1 = 3: DDE-Trigger über HotKey-Poke
prm1 = 4: Communication Port Trigger (Port must be open for input)
type = 2: Software Line Trigger: prm1 = horz/vert = 0|1, prm2 = Länge der Linie
type = 3: Software Line Trigger: Setze Line Start: prm1 = x, prm2 = y
type = 4: Software Line Trigger: prm1 = Threshold, prm2 = Trigger on low/high (0|1)
type = 5: Setze minimum Delay between 1st and 2nd Trigger in [ms] (prm1). Hierdurch kann sichergestellt werden, daß nicht ein Prellen zum direkten Abbruch-Trigger führt.
type = 6: Setze Measurement Time [ms] (prm1), und Time Start:
prm2=0: Start Time with 1st image, prm2=1: Start Time at Trigger
type = 7: Setze Trigger Delay: prm1 = Start Delay, prm2 = Stop Delay [ms]
type = 8: Setze Hardware Trigger-Line: 100..199 Acqisition-Karte, 0..47 auf Digital-I/O (z. B. DII Decision Computer Timer/Counter card), -1 = no change

type = 9: prm1 = DDE or COM-port Ascii Character for Trigger, -1 for all characters, -2 for don't change
prm2: Hot-Key ESC or 'E', or CR or 'A' can always trigger real time functions or ring buffer acquisitions. This also works with DDE or TCP/IP interface even if those are not enabled. Set prm2 to 0: clear trigger character that was set before. (use prm1=-2)
type = 10: Get oder Set Hardware-I/O-Line: prm1 = get_set: -1 = get, 0 = set0, 1 = set1 prm2 not used, vielleicht Hardwareabhängig!. Achtung: bei DII-Karte wird beim ersten Ansprechen die I/O-Line auf Input (bei get) oder Output (bei set) gesetzt. prm = -1: no change
type = 11: prm1 = 1: Set Immediate Trigger, Functions start immediately without Trigger, resets automatically to 0 after function execution.
type =12: Setze Hardware Check-Line:
if (prm1>=0 && prm2>=0) Check (prm1=0|1: off|on) Ext.Trigger I/O-Line (prm2=0..n), write status of I/O-Line to DDE or file during 2D/3D-Tracking
prm = -1: no change
type=13: if (prm1>=0 && prm2>=0) Set synchronization of stereo trigger to make sure two frame grabbers start with the same image (prm1=0|1), use Sync-I/O-Line (prm2=0..n), n=0..47 on DII-Card
prm = -1: no change
type=14: prm1 = 0..n: Set Stereo-synchronization Trigger Delay in [us] after Rising Edge of I/O -> Acquisition. prm1 = -1: no change

Spezielle implementierte Echtzeitfunktionen:

rt_position	4, B / I	type, param1, param2, param3: integer; Echtzeit-Positionsbestimmung und Marker- oder Objektverfolgung (siehe 2D/3D-Positionsbestimmung)
rt_trpiv	4, B / I	type, param1, param2, param3: integer; Echtzeit-PIV, Time-Resolved-Particle-Image-Velocimetry (siehe PIV)
rt_savestream	4, B / I	interactive, namecount, numbercount, numberstart: integer; Abspeichern eines Video-Streams in Echtzeit. (Zusatzmodul Echtzeitverarbeitung, siehe auch Speicherfunktionen)

3D-Rekonstruktion (Manual Kapitel 8.12. und 14. und Zusatzmodul)

pos3d_setup 2, B/ID

type, param: integer; Einstellung aller Parameter und Berechnung der Transformationsmatrizen zur 3D-Positionsrekonstruktion. Es gibt grundsätzlich drei Möglichkeiten der 3D-Rekonstruktion: Entweder sind die genauen Positionen und Blickwinkel der beiden Kameras bekannt und werden hier eingegeben, woraus dann zwei Transformationsmatrizen berechnet werden können, mit denen dann jeder beliebige Raumpunkt aus seinen zwei Projektionsabbildungen ermittelt werden kann, oder es sind genau 6 Raumpunkte exakt bekannt und deren Abbildungen in beiden Teilbildern der beiden Kameras sichtbar, so daß sie mit der Objekt-Tracking-Funktion bestimmt werden können. Aus diesen 12 Abbildern der 6 Raumpunkte in beiden Kameraansichten können wiederum die Transformationsmatrizen bestimmt werden, sodaß alle folgenden Raumpunkte rekonstruiert werden können. Oder es sind schließlich mehr als 6 Kalibrierpunkte gegeben, so daß über eine Least Square Gauss Berechnung sehr genau kalibriert wird.
Außerdem kann eine Absolutlage des zu vermessenden Objektes eingegeben werden oder die Art des benutzten Drehungssytemes festgelegt werden. Weiterhin gibt es einen Reset aller 3D-Daten, das Setzen einer Referenz-Objektlage, die Wahl der 3D-Ausgabe in Punktkoordinaten oder in Objektlage und Drehung, in unterschiedlichen Drehungssystemen.
Schließlich kann hier auch eine Korrektur von Daten einer Oberflächen-verbiegungsmessung (siehe FCM-Zusatzmodul) initialisiert werden.
type=-1: Reset Position: Zurücksetzen der Systeminitialisierung der Null-Ausgangsposition. Die nächste Positionsbestimmung wird dann als Referenzmessung verwendet und initialisiert intern alle Variable. Diese Funktion kann benutzt werden, um bei leicht geänderten Markerpositionen, z. B. durch geänderte Beleuchtung oder geringfügig geänderte Versuchsbedingungen wieder eine definierte Ausgangslage zu erhalten. Eine Objektlage wird grundsätzlich relativ zu einer Ausgangslage angegeben, die durch die Markerpositionen beim ersten Tracking festgelegt wird. Wenn beim ersten Tracking Fehler auftreten, so kann ebenfalls diese Reset-Funktion zur Korrektur benutzt werden.
type=0: Interaktive 3D-Positionsparameterbestimmung: Gehen Sie folgendermaßen vor:
Entweder: Eingabe der Kamerapositionen:
- Schalter "Set Camera Positions" einschalten und Camera-Dialogbox öffnen.
- x,y,z-Positionen für Kameras 1 und 2 eingeben
- dann x,y,z-Positionen für den Blickpunkt (View Point oder Center of Interest)
- dann die Rotation um die optische Kameraachse und Kalibrierfaktoren
- Transformationsmatrizen berechnen
Oder: Eingabe von 6 Referenzpunkten und 12 Markerpositionen:
- Schalter "6 Point Rekonstruction" einschalten
- alle 6 Raumpunkte eingeben mit x,y,z-Koordinaten
- Schalter "Actualize Calibration Points" betätigen. Hierzu muß man zuvor die 12 Markerpositionen mit der Funktion "Define Markers" [rt_position(0, marker_size, marker_shift, 12)] interaktiv oder per MACRO eingeben.
- Transformationsmatrizen berechnen.
Oder: Eingabe von mehr als 6 Referenzpunkten mit mehr als 12 Markern:
- Schalter "Least Square Rekonstruction" einschalten
- alle Raumpunkte eingeben mit x,y,z-Koordinaten
- Schalter "Actualize Calibration Points" betätigen. Hierzu muß man zuvor die Markerpositionen mit der Funktion "Define Markers" [rt_position(0, marker_size, marker_shift, nn)] interaktiv oder per MACRO eingeben.
- Transformationsmatrizen berechnen. Fehlervektoren werden angezeigt.
Die komplette Einstellung aller Variablen und Schalterstellungen kann mit der "Save All"-Schaltfläche nach Eingabe eines Filenamens (Extension .P3D wird automatisch angehängt) gesichert werden und später mit der "Load All"-Schaltfläche wieder hereingeladen werden. Einzelne Referenzobjekte können ebenfalls Abgespeichert und wieder hereingeladen werden, hier im ASCII-Format mit der Extension .O3D, so daß man diese Objekt-Files auch mit einem beliebigen Editor ändern kann. Ohne Einschränkung der Allgemeinheit werden die Transformationsmatrizen so bestimmt, daß eine Projektion auf die xy-Ebene (z=0) angenommen wird. Es existiert eine alternative Eingabemöglichkeit, die z. Zt. jedoch nur für Testzwecke interaktiv vom picCOLOR-Programm aus zur Verfügung gestellt wird: Über eine Anfangstranslation, beliebige Drehungen um die x,y,z,y,x-Achse in dieser Reihenfolge, eine Endtranslation und eine anschließende Projektion auf eine der Ebenen (x=0, y=0, oder z=0) sowie einen beliebigen Projektionspunkt auf der jeweiligen zur Projektionsebene senkrechten Achse

		ist damit möglich. **type = 1:** Interaktive Eingabe der absoluten x, y, und z-Position und der Winkel um Gier-, Nick- und Roll-Achse des zuletzt bestimmten Objektes mit 3 Raumpunkten (von denen keiner im Koordinatenursprung liegen sollte). Aus diesen Absolutkoordinaten und Winkeln werden additive Differenz-Werte gebildet, die dann zu allen folgenden Objektlagen hinzuaddiert werden. **type = 2:** Festlegung des Drehungssystemes und der Art der Datenausgabe: param=0: Ausgabe der Ergebnisse nur mit den 3D-Punktkoordinaten param=1:Drehungssystem ist das yaw/pitch/roll-System. Die Ausgabe der Objektlage mit x,y,z und yaw, pitch, roll param=2:Drehungssystem ist das Euler-Winkelsystem mit Nutation (theta), Präzession (psi) und Drehung (phi). Die Ausgabe der Objektlage mit x,y,z und theta, psi, phi param=3: Drehungssystem ist das alpha/beta/gamma-System. Ausgabe der Objektlage (Ebenenlage) mit x,y,z und alpha, beta, und gamma (Winkel des Ebenen-Normalenvektors zu den Achsen) param(Bit16)=0: (param = 0x0nnnn) Append WorstError , TimeCode only param(Bit16)=1: (param = 0x1nnnn) Append status with 8 (or old 6) values to 3D-data: WorstError , RingBufferDelay, I/O-Status, FrameIndex,TimeCode1, TimeCode2, LostFrames1, LostFrames2 param(Bit17)=1: (param = 0x2nnnn) Save additional 3D-Coords to file, no matter how the rotation system is defined. param(Bit18)=0: (param = 0x4nnnn) Save normal 8 value status line with 3D-data to file, 1: Save old version 6 value status line (worsterror, delay, io-status, frameindex, timecode, lostframes) param = -1: Rückgabe: IVAR[10] = Pos3d.Type3d (0|1|2|3 = 3D, YPR, Euler, abc, +status, +savealways) **type = 3:** Eingabe der Parameter für eine Korrektur von Oberflächenverbiegungsdaten (FCM-Modul) in einer Dialogbox. **type = 4:** Art der Korrektur bei der FCM-Oberflächenanalyse: param=-1: Rückgabe: IVAR[10] = pgm.usepos param=0: keine Korrektur der FCM-Daten. param=1: benutze dz, pitch und roll für die FCM-Korrektur param=2: benutze nur die vertikale Translation dz für die FCM-Korrektur param=3: Interaktiver Input der Korrektur-Ebenengleichung mit Ordinate an linker oberer Ecke der ROI und den Gradienten in x-und y-Richtung. **type = 5:** param = 0|1|2 = caminput|6point|leastsquare (>=7 points!) param=-1: IVAR[10] = type of reconstruction method **type = 6:** calculate position matrix **type = 7:** Iterate calibration, using actual result points **type = 8:** param = ref=0 | result=1: Transfer Ref/Res-Data-Set to Vector Field **type = 9:** param = ref=0 | result=1: Transfer Ref/Res-Data-Set from Vector Field **type = 10:** param = -1: show differences: reference-result points. These differences are the error vectors of the least-square calibration, if directly after calibration a first measurement is performed.
pos3d_plot	4, B	num, set, left_right_both, col: integer; Graphische Darstellung der 3D-Positionen als 2D-Projektionen (auf Teilbildern/Vollbildern). num: Anzahl der Raumpunkte, die dargestellt werden sollen. Für jeden Raumpunkt werden zwei Abbildungen dargestellt, d. h. für beide Kamerabilder. set = 0|1: 0 ist das Referenz-Set, das zur Erstellung der Transformationsmatrizen benutzt worden war. set=1 ist das Benutzer-Set, d. h. hier werden die Koordinaten der jeweils letzten 3D-Rekonstruktion abgelegt. left_right_both = 0|1|2: es werden nur die 2D-Projektionen für linke Kamera (Cam1), rechte Kamera (Cam2) oder auch für beide Kameras dargestellt. col ist die Farbe (Grauwert/Pixelwert), in der die Punkte dargestellt werden sollen. Bei col=-1 werden die Punkte auf dem Bildschirm nur invertiert und kein Bildspeicher ist danach aktiviert. Bei col=-2 werden die Punkte in den Pixelwerten 1,2,3,.... dargestellt, so daß bei sinnvoller Farbtabellenbelegung (First16 in der Look-Up-Table-Dialogbox) eine sehr gute Unterscheidung möglich ist. Bei allen anderen Werten von col [0..255] kommen die jeweiligen Grauwerte zur Darstellung
pos3d_calc	3, B/IM	num, single_continuous, extrapolationrun: integer; Berechnung der 3D-Positionen für "num" Raumpunkte aus 2*num Bildprojektionspunkten (Markerpositionen) mit den zuvor definierten Transformationsmatrizen, wenn die Bildpunkte mit der Funktion "rt_position(5,0,0,number)" in die Positionsbestimmungsvariablen transferiert wurden.

		Die Ergebnisse werden angezeigt oder bei geöffnetem Datenfile auf diesen geschrieben. Wenn Headerzeilen eingeschaltet sind, so wird wie folgt ausgegeben: *31* x y z yaw pitch roll worsterror (oder: theta psi phi) ..... Auf Wunsch können auch die tatsächlichen Koordinaten der gefundenen Raumpunkte ausgegeben werden. Dies erfolgt dann in der folgenden Form: *31* Point, x, y, z WorstError (3D-Position) 1 x[1] y[1] z[1] 0 2 x[2] y[2] z[2] 0 ..... num x[num] y[num] z[num] Es werden zusätzlich ErrorCode, TimeCode und eventuell noch weitere Informationen angegeben. num: Anzahl 3D-Punkte, also Anzahl der Marker in beiden Bildern durch 2. single_continuous = 0|1: 0: result output immediately, 1: sequence, no output extrapolationrun = 0|1: 0: normal execution; 1: reconstruct all correctly recognized error free markers only and do not output anything.
pos3d_set	8, B	ref_meas, n: integer, component, value: float; Eingabe von Referenzpunkten. ref_meas = 0: Set Reference Point[n]. For return values use Pos3d.markercount ref_meas = 1: Set Measurement Point[n]. For return values use Rt.count/2 Return values: IVAR[10]=old point count, [11]=new point count n = [0..MAXMARK/2) component = x=1 | y=2 | z=3. value double precision point coordinate
pos3d_get	2, B	ref_meas_posangle, n: integer; Abfragen der eingegebenen Referenzpunkte oder der ermittelten 3D-Punkte oder der letzten ermittelten Objektlage und -drehung. ref_meas_posangle = -1: Get ErrorVector of Point[n] nach FVAR[10]=x,[11]=y, [12]=z ref_meas_posang = 0: es werden die Referenzpunkte ausgelesen ref_meas_posang = 1: es werden die gemessenen Punkte ausgegeben n: Nummer des auszugebenden 3D-Punktes. Die Werte werden in den Floating-Point-Variablen übergeben: FVAR[10] = x, FVAR[11] = y, FVAR[12] = z ref_meas_posang=2: Ausgabe der Objektlage mit x,y,z-Position und den Drehungen um Yaw, Pitch, und Roll-Achse, oder Euler-Winkel. FVAR[10] = x, FVAR[11] = y, FVAR[12] = z FVAR[13] = yaw/theta, FVAR[14] = pitch/psi, FVAR[15] = roll/phi. Wenn in diesem Fall n!=0 ist, werden die Werte in Messagebox angezeigt. Bei ungültiger Eingabe wird als ReturnValue -1 zurückgegeben und man sollte dann die in IVAR[10] übertragene maximale Anzahl von Markern überprüfen.
pos3d_saveload	3, B/I	save_load, prm_refobj_measobj, interactive: integer; Abspeichern oder Hereinladen aller Parameter zur 3D-Positionsbestimmung oder eines 3D-Referenzobjektes. Außerdem Abspeichern oder Hereinladen eines Objektes für die 3D-Objektlagevermessung, entweder eines Referenzobjektes oder eines Meßobjektes. Die Parameter sind die eventuell bekannten Kamerapositionen, die Referenzpunkte, die Projektionsorte der Referenzpunkte in zwei Teilbildern, und schließlich die Koeffizienten der Transformationsmatrizen. prm_refobj_measobj = 0: Abspeicherung und Laden aller Parameter. Es wird ein Binärfile mit der Extension .P3D (für Parameter-3D) geöffnet. prm_refobj_measobj = 1: Abspeichern und Laden der Referenzpunkte eines Referenzobjektes. Es wird ein ASCII-File mit der Extension .O3D (für Objekt-3D) geöffnet, der auch mit einem beliebigen Editor bearbeitet werden kann. prm_refobj_measobj = 2: Abspeichern und Laden der Referenzpunkte eines Meßobjektes. Hiermit kann z. B. ein Meßßobjekt definiert werden und dann in die 2D-Markerkoordinaten umgerechnet werden, so daß man sich die interaktive Mauseingabe spart. Returnwert beim Laden des Referenz- oder Meßobjektes: IVAR[10] = n (Punktanzahl) ReturnValue ist 1 bei erfolgreicher Ausführung, 0 bei falscher (leerer) Filenameneingabe, und -1 bei Fehler (File not Found oder Schreib/Lesefehler)
pos3d_transform	7, B	type: integer; value: float; Transformieren eines 3D-Punktes type = -2: Reset all values type = -1: show interactive dialog box

type = 0: do rotation
type = 1: input translation in x axis
type = 2: input translation in y axis
type = 3: input translation in z axis
type = 4: input rotation about x axis (roll)
type = 5: input rotation about y axis (pitch)
type = 6: input rotation about z axis (yaw)
type = 7: input 2nd rotation about y axis
type = 8: input 2nd rotation about x axis
type = 9: input 2nd translation in x axis
type = 10: input 2nd translation in y axis
type = 11: input 2nd translation in z axis

Optical Character Recognition (Kapitel 8.13. und 14. und Zusatzmodul):

ocr_search	3, B/I	display_buffer, min_affinity, savedata: integer; Erkennung von zuvor eintrainierten Zeichen. display_buffer = -1 | 0 | 1..n = quiet | show affinity | show chars in buffer # 1..n min_affinity = 0..100%, set the minimum affinity that characters will be accepted. Useful is some 70..80% minimum affinity savedata = -1 | 0 | 1 = interactive | no | save
ocr_params	8, B	type, iparm: integer, dparm, dummy: float; Setzen von OCR-Parametern type = 1: OCR_Rot.rhomin = dparm, Minimales Rho zu Annerkennung eines Wortes type = 2: iparm = 0: use max length word(s) only (averaged, if more than one) iparm = 1: average all lines
ocr_train	4, I	type, dx, dy, n: integer; Trainieren von Zeichen zur Erkennung
ocr_saveload	2, B/I	save_load, interactive: integer; Speichern oder Laden von trainierten Zeichensätzen.
ocr_lineangle	8, B/I	Bestimme den Winkel von Schriftzügen. Hierzu müssen die Objekt-Such-Parameter richtig gesetzt sein, damit die einzelnem Zeichen sicher erkannt werden, i.e. dunkel/hell, Grauwertschwelle, min/max Größe, usw... Results: IVAR[10] = wordcount IVAR[11] = length of word with maximum rho IVAR[12] = count of words used for average FVAR[10,11,12,13,14] = Rho, a, b, c, d FVAR[15] = angle of word with maximum rho (savest word) FVAR[16] = mean angle of all used words

BarCode Recognition (Manual Kapitel 8.14. und 14. und Zusatzmodul):

bar_code 4, B/I

inter, type, prm1, prm2: integer; Barcode-Auswertung, z. Zt. sind folgende Codes implementiert: EAN8 und EAN13, UPC-A und UPC-E, Code39, 2 aus 5 interleaved
interactive = 1: type = 1: angle method, type = 2: fft method (not tested)
interactive = 0:
type = -1: Request BarCode Data Set[prm1]
IVAR[10] = sxm, IVAR[11] = sym
IVAR[10] = sx1, IVAR[11] = sy1
IVAR[10] = sx2 , IVAR[11] = sy2
IVAR[10] = sx 3, IVAR[11] = sy3
IVAR[10] = sx 4, IVAR[11] = sy4
FVAR[10] = Angle, FVAR[11] = Count
STRING[1] = CodeString
type = 0: set all Parameters to Default Values
type = 1: decode barcode, search angle method, prm1 = 0|1 = single|all codes
type = 2: decode barcode, search fft method (not tested...), prm1 = 0|1 = single|all codes
type = 3: Max Tile Size = prm1, Overlap = prm2
type = 4: GaussFilter of Quarter Image = prm1 = 0|3|5|7, ReadCutCodes = prm2 = 0|1|2|3, try reading codes cut by ROI or image border. Bit1: Set: try reading cut codes, Bit2: Set: don't care for stop sign.
type = 5: SobelNeighbor = prm1, SobelThreshold = prm2
type = 6: HistoFilter = prm1, HistoIter = prm2
type = 7: HistoRange = prm1, MorphKernel = prm2
type = 8: MorphClose = prm1, MorphOpen = prm2
type = 9: AreaMin = prm1, AreaMax = prm2
type = 10: LineWide = prm1, LineGrad = prm2 = 3|5|7
type = 11: LineFilter = prm1, MinContrast = prm2
type = 12: CodeType = prm1, EdgeCount = prm2
type = 13: EdgeStop = prm1, EdgeFilter = prm2
type = 14: VerySmallCode = prm1= 0|1|2: 0: Normal large codes, search angle +-5, +-10 degree in center line, 1: small codes, search angle +-1, +-2 degree in center line, 2: very fine codes, search angle +-1 degree also at up/down lines, and additionally try (linefilteriteration-1), and try zoom-in by factor of 2, gauss(3), and all possible search lines. prm2 is the width of the quiet zone in pixel sizes (default=7)
type = 15: prm1 = morph gray close = open (aftzer sobel-direction).
Alle Werte < 0 werden nicht geändert
Wenn kein Stop-Zeichen gefunden wird, wird 0 zurückgegeben.

barcode2d 4, B/I

inter, type, prm1, prm2: integer; 2D-BarCode-Erkennung, z. Zt. sind folgende Codes implementiert: PDF417.
type = -2: Free PDF-Table
type = -1: Request BarCode Data Set[prm1]
IVAR[10] = sxm, IVAR[11] = sym
IVAR[10] = sx1, IVAR[11] = sy1
IVAR[10] = sx2 , IVAR[11] = sy2
IVAR[10] = sx 3, IVAR[11] = sy3
IVAR[10] = sx 4, IVAR[11] = sy4
FVAR[10] = Angle, FVAR[11] = Count, STRING[1] = CodeString
type = 0:set all Parameters to Default Values
type = 1: decode PDF417
type = 2: reserved for different code type
type = 3: Max Tile Size = prm1, Overlap = prm2
type = 4: GaussFilter of Quarter Image = prm1 = 0|3|5|7, prm2 not used
type = 5: SobelNeighbor = prm1, SobelThreshold = prm2
type = 6: HistoFilter = prm1, HistoIter = prm2
type = 7: HistoRange = prm1, MorphKernel = prm2
type = 8: MorphClose = prm1, MorphOpen = prm2
type = 9: AreaMin = prm1, AreaMax = prm2
type = 10: LineWide = prm1, LineGrad = prm2 = 3|5|7
type = 11: LineFilter = prm1, MinContrast = prm2
type = 12: CodeType = prm1, EdgeFilter = prm2
type = 13: DilateX = prm1, DilateY= prm2
type = 14: Global/Dynamic = prm1
Alle Werte < 0 werden nicht geändert

Neural Network

(not yet implemented into MACRO language)

Graph Theory:

voronoi	3, B	generators, output, graphics: integer: Erzeugung von Voronoi und Delaunay Graphen: generators: benutze entweder Random Dots = 1..n, oder picCOLOR Objects = -1. output = 0|1: write to file graphics: bit0 = Voronoi Graph 1 bit1 = Voronoi Graph 2 bit2 = Delaunay Graph Achtung: preliminary function, may crash!!!.

Testbilderzeugung (Manual Kapitel 6.9.):

normtest	3, B	gridsize, color, basic: integer; Erzeugung eines Gitter-Testbildes zur Vermessung von Bildern oder geometrischen Größen. gridsize: Abstand der Gitterlinien, bei gridsize==0 wird die Gitterweite automatisch zu 1/8 der Bildgröße in x- und y-Richtung gewählt. color: 0..255: Grauwert für die Gitterlinien basic: 0|1: Bei ‘‘basic’’==0 werden nur Gitterlinien in x- und y-Richtung gezeichnet. Bei ‘‘basic’’==1 werden zusätzlich ein großer Kreis und eine Grau-Rampe gezeichnet.
gauss_noise	5, B	sigma: float; Add Gaussian Noise, 1.0 <= sigma <= 255.0
sin_wave	8, B	contrast, orientation: integer; phase, freq: float; Erzeugung von Sinus-Schwingungen 0 <= contrast <= 255 0 <= orientation <= 360, 0.0 <= phase <= 360.0, 0.0 < freq < MAX, wobei MAX nicht größer als die ROI-Breite oder Höhe sein sollte.
slope	3, B	start, contrast, orient: integer; Zeichnen eines Graukeils: start = Startgrauwert contrast = Grauwertumfang über die gesamte ROI-Breite orient = Winkel des Graukeils
specle_noise	5, B	proportion : float; Erzeugen eines Testbildes mit Specle-Rauschen 0.0 <= proportion <= 1.0
random_seed	3, B	distribution, number, color: integer; Erzeugen einer zufälligen Verteilung von Punkten: distribution = 1|2: 1=uniform, 2=normal number: 1..n Punkte color: 0..MaxGrau.

Graphik- und Texteinbindung (Manual Kapitel 10.):

imagepaint	0, ID,	kein Parameter; Interaktives Graphikmodul. Return_Value immer 1.
draw_params	4, B	type, prm1, prm2, prm3: integer; Setze Graphikparameter: type = 0; Setze Fadenkreuz 0: x0 = prm1, y0 = prm2 type = 1; Setze Fadenkreuz 1: x1 = prm1, y1 = prm2 type = 2; Setze Fadenkreuz 2: x2 = prm1, y2 = prm2 type = 3; Setze Textzeiger: xtext = prm1, ytext = prm2 type = 4; Setze Textfarben: Foreground = prm1, Background = prm2 type = 5; Setze Zeichenfarben: Foreground = prm1, Background = prm2, for internal drawing, used for Linux line drawing Auf -1 gesetzte Textfarben bedeuten Transparenz. Wenn beide Werte auf -1 gesetzt sind, wird invertiert und mit transparentem Hintergrung gezeichnet. Return Value ist immer 1.
draw_vector	3, B	linewidth, color, vect_head: integer; Zeichne Linie oder Vektor: linewidth ist die Breite in Pixeln. Achtung: aufgrund des internen Algorithmus kann es bei bestimmten Winkeln der Linie dazu kommen, daß transparente Pixel entstehen. color kann jeder Grauwert von 0..255 sein. vect_head wählt einen bestimmten Vektorkopf aus. Dies ist jedoch noch nicht implementiert. Die Linie wird von Fadenkreuz 1 nach Fadenkreuz 2 gezeichnet. Hierzu müssen die beiden Fadenkreuze zuvor gesetzt werden, was mit set_two_cross(x1,y1,x2,y2) oder draw_params(1,x1,y1,0) und draw_params(2,x2,y2,0) getan werden kann. Return Value ist 1, außer wenn die Linienlänge gleich 0 oder > MAXLINE ist.
draw_ellipse	4, B	major, minor, angle, color: integer; Zeichnen einer Ellipse, Zentrum ist Fx0/Fy0
draw_circle	4, B	w, h, yleft<<16|xtop, color: integer; Zeichnen eines Kreises im angegebenen Rechteck
flood_fill	2, B/I	interactive, color: integer; ‘‘Floodfill’’, d. h. Füllen einer Fläche mit der Farbe/Graustufe ‘‘color’’. Interaktiv wird mit einem Fadenkreuz ein Punkt eingefragt, von dem ab bis an die Grenzen der aktuellen Farbe/Grauwert gefüllt wird. Nichtinteraktiv wird die aktuelle Position des einfachen Fadenkreuzes benutzt.
brush_stamp	2, I	mode, destinationbuffer: integer; Use brush/stamp-Graphic tool. mode = 0: brush mode = 1: stamp. If destinationbuffer=0, use same buffer. If destinationbuffer>0, stamp will work in that destination buffer. (if shift not 0, ask whether this was intended)
retouch_radius	2, B	interactive, radius: integer; Setzen des Radius' für das brush/stamp-tool. radius in [0..16]. radius = 0 means 1 Pixel
sharp_smooth	1, B	mode: integer; Einstellung des brush/stamp-tools auf scharfe Ränder oder (1+cos)-förmige weiche Ränder.
rub_add_mul	3, B	interactive, mode, const: integer; Setzen von Parametern für das brush/stamp-tool: mode = 0: use rubber mode = 1: add const to image. const in [-255..255] mode = 2: mul const to image. const in [0..1000] (const will internally be divided by 100, giving a mul-factor of 0.0 to 10.0)
draw_text	9, B	text: String; Zeichnet den angegebenen String and die Stelle des zuvor eingegebenen Textzeigers (mit draw_params(3,x,y,0) mit den definierten Vorder- und Hintergrundgrauwerten (draw_params(4,Fcolor,Bcolor). Auf -1 gesetzte Farbe bedeutet ‘Transparent’. Wenn beide Werte auf -1 gesetzt sind, wird invertiert und mit transparentem Hintergrung gezeichnet. Return Value ist immer 1.
open_font	9, B/IM	filename: String; Öffnen und Einlesen eines der im PICCFONT liegenden Textfonts. Kann durch Leerstring ‘‘’ auf den internen Systemzeichensatz (Größe 8*8) zurückgesetzt werden.
surface_3d	8, B	interactive_stereo, surface_grid_width: integer; zscale, lightangle: float; 3D-

		Oberflächendarstellung von Grauwertgebirgen durch Interpretation des Grauwertes als Höhe, in Flächen- und Gitter-Darstellung. (default zscale = 0.1). interactive_stereo: -1: interactive=1, no stereo 0: no interactive, no stereo, just plot 1: interactive=0, stereo=1, plot surface_grid_width: <0: surface display, >0 (1..n) grid display with grid width 1..n zscale: Skalierung in der Z-Achse. (gray value to height conversion: -10.0..10.0) lightangle: Lichtquelle aus, wenn lightangle==0.0, sonst eingeschaltet lightangle: Winkel der Lichtquelle von der x-Achse um die y-Achse [0..180] 0.0<lightangle<180.0: light angle in x-direction around y-axis (surface only)
set_eyepoint	7, B/I	inter_xyz: integer; eye_xyz: float; Setzen des Beobachterpunktes (eye_point) und Bildschirmabstandes für die 3D-Oberflächendarstellung. inter_xyz = -1: interaktiv mit Dialogbox (alle anderen Werte nur in der Dialogbox) inter_xyz = 0: Setzen des Bildschirmabstandes für Stereodarstellung inter_xyz = 1,2,3: Setzen des x,y,z-Augenpunktes inter_xyz = 4,5,6: Setzen des x,y,z-Center of Interest inter_xyz = 7,8,9: Setzen des x,y,z-User-Center of Interest inter_xyz = 10: Setzen der Eye Rotation eye_xyz: x,y,z oder Bildschirmabstand oder anderer Parameter, abhängig von inter_xyz
vect_3d	4, B/I	inter, stereo, vectortype, color: integer; Darstellung von 3D-Vektordatensätzen, eventuell animiert. (siehe auch Kapitel Measuerement) inter = 0: nicht interaktiv inter = 1: animierte 3D-Darstellung über Maussteuerung inter = 2: interaktive Eingabe des Beobachterstandpunktes und anderer Parameter. stereo = 1: es wird versucht, nichtinteraktiv ein Stereobild zu erzeugen. Wenn kein Stereobildspeicher definiert ist, wird in die Rot/Grün-Speicher eines Farbbildspeichers geschrieben und nachher ein gedithertes Farbbild angezeigt, daß mit einer Rot/Grün-Brille zu betrachten ist. vectortype = 1: alle Vektoren werden miteinander verbunden. vectortype = 2: es werden Vektorpfeile dargestellt. Hierbei werden die Punkte 1,2 als Anfangs- und Endpunkte eines Vektors benutzt, dann 3,4, dann 5,6 usw. Die Zeichenfarbe color kann zu 0..255 und außerdem noch mit -1 als invertierend eingestellt werden. Die Funktion Zeichnet nur auf den Bildschirm, man muß also eventuell noch mit der Funktion scrn_to_buf_ROI in einen Bildspeicher kopieren.
get_pix	2, B	x,y: integer; Auslesen eines Bildpunktes auf dem Bildschirm (nicht im Bildspeicher). Der Wert wird in IVAR[10] übergeben.
put_pix	3, B	val, x, y: integer; Schreiben eines Bildpunktes auf dem Bildschirm (nicht in den aktuellen Bildspeicher)

Anhang: Fehlermeldungen

Fehlermeldungen beim Laden des MACRO-Programmes:
"Temporary memory could not be allocated"
"Macro Load ERROR!"
"Macro file too long: truncated"
"No memory for MACRO"
"No MACRO program loaded"
Macrofunktion wurde nicht gefunden oder paßt nicht den den dafür vorgesehenen Speicher. Z. Zt. kann ein Macro-Programm incl. aller Kommentare bis zu 64kByte lang sein. Dies wird auf Wunsch erweitert. Nach einer Vorkompilierung wird das Macro-Programm dann auf den Heap-Speicher geschoben, der jedoch auch begrenzt sein kann.

"MACRO too long for Notebook"
Für Testzwecke kann das Macro nach der Kompilierung in den Notebook-Speicher geschrieben werden, der nur eine begrenzte Länge hat.

"Function or Variable does not exist: name"
"Function or Variable does not exist: name: Continue Anyway (OK)?"
"Command not implemented"
"Command not implemented for DDE or TCP/IP interface"
"Unknown command"
Die angegebene Funktion wurde nicht in der Command- oder Funktionstabelle gefunden. Überprüfen Sie die Schreibweise und die Rechtschreibung! System-Kommandos müssen immer groß, Funktionsnamen müssen genau wie in der Macro-Funktionstabelle angegeben geschrieben werden, d. h. meist in Kleinschreibung.Für die Übertragung von Kommandos über die DDE oder TCP/IP-Schnittstelle sind nicht alle picCOLOR-Macro-Funktionen zugelassen.

"Unrecoverable ERROR! At Function: F@nnn = function_name; Parameters: xxxxx"
Fehlerabbruch - eine Funktion lieferte eine "-1" als Return-Value und zeigt dadurch eine Fehlerbedingung an. Dies kann aus verschiedenen Gründen geschehen: Speichermangel, Benutzerabbruch während der Funktionsausführung mit der ESC-Taste, usw. Dieses Verhalten kann über die Funktion "stop_on_error" geändert werden. F@ ist dabei die Nummer der Funktion in der internen Tabelle. Variablen in der Parameterzeile mit einem Index >= 256 sind definierte Variable mit Namen, der leider bei der Kompilation aud Platz- und Geschwindigkeitsgründen nicht abgespeichert wird.

"Warning! Function may be not completed!"
Kein Fehler, nur eine Warnung: normalerweise liefern Funktionen bei normalem Abschluß eine 1 zurück, bei Fehlerabbruch eine -1. Unter bestimmten Bedingungen und bei bestimmten Funktionen kann jedoch auch eine nicht fehlerhafte Beendigung einer Funktion vorkommen, z. B. durch eine abschlägige Benutzerentscheidung. Dann liefert sie eine 0 zurück und es wird die Warnung angezeigt, die mit der Funktion "stop_on_error" unterdrückt werden kann.

"MACRO aborted on user interrupt!"
"Macro ends by STOP command"
Die Ausführung der Macrofunktion wurde abgebrochen.

"Warning: missing parameter"
"Wrong parameter: xxx"
"Not a valid String"
Die Art und Anzahl der übergebenen Parameter für Funktionen muß dem Handbuch oder im Zweifelsfall der Macro-Funktionsliste im Programm entnommen werden. Nicht angegebene Parameter werden in der Regel zu "0" angenommen.

"':=' expected"
Bei einer Zuweisung wurde nur ein einzelnes Gleichheitszeichen '=' benutzt. Dieser Fehler führt nicht zum Abbruch.

"Division by 0"

Bei einer Division wird versucht, durch 0 zu teilen. Dies kann z. B. vorkommen, wenn ein Divisionsparameter nicht richtig eingelesen werden kann, weil z. B. ein Variablenindex falsch angegeben ist und dann 0 angenommen wird.

"Wrong Operator"
Falscher Operator bei der Zuweisung. Erlaubte Operatoren sind in der Sprachbeschreibung aufgelistet. (siehe auch Help-Screen)

"FOR-loop count < 0 !"
Der loop count muß >= 0 sein!

"Wrong lexical level at 'FOR'"
"Wrong lexical level at 'WHILE'"
Ein FOR oder WHILE wurde an einer nicht zugelassenen Stelle entdeckt - das kann z. B. auch daran liegen, daß ein entsprechendes ENDFOR oder ENDWHILE fehlt.

"Too many 'FOR'-loops"
"Too many 'WHILE'-loops"
"Too many nested 'FUNCTION' calls"
z. Zt. sind 16 ineinandergeschachtelte FOR-loops und weitere 16 ineinandergeschachtelte WHILE-loops erlaubt. Auch Funktionsaufrufe düerfen nicht mehr als 16-fach verschachtelt sein.

"Unexpected 'ENDFOR'"
"Unexpected 'ENDWHILE'"
Ein ENDFOR oder ENDWHILE wurde an nicht zugelassener Stelle gefunden.

"Wrong comparison operator "
Als Vergleichsoperatoren sind erlaubt: =, #, <, <=, >, >=.

"Missing 'ELSE'"
"Missing 'ENDIF'"
"Missing 'FUNCTION'"
"Missing 'LABEL'"
"Missing ENDFOR"
"Missing ENDWHILE"
Eines der angegebenen Kommandos wurde im Zusammenhang erwartet und konnte nicht gefunden werden. Wichtig: ein IF muß immer eine ELSE-Anweisung haben, die mit der ENDIF-Anweisung abgeschlossen wird.

"Running into FUNCTION: use STOP"
Wenn Funktionen programmiert werden, so muß das Macro-Programm mit einer STOP-Anweisung beendet werden. Funktionen müssen am Ende nach dieser STOP-Anweisung stehen.

"RETURN without Function-CALL"
"Cannot RETURN from main MACRO thread"
Eine RETURN-Anweisung wurde gefunden, ohne daß eine zugehörige FUNCTION-Anweisung vorhanden ist.

"Error At Function Call: Return Address In Use"
Dieser Fehler sollte eignetlich nicht vorkommen... Bitte reproduzieren und MACRO einschicken!

"'GOTO' not allowed today"
"'LABEL' not allowed today"
Die Kommandos GOTO und LABEL sind in der derzeitigen Implementierung nicht erlaubt. Das Heraussprungen und noch vielmehr das Hineinspringen in eine Schleifenstruktur würde zu sehr schwierigen Programmablaufproblemen führen. So wäre z. B. ein Schleifenzähler beim Hineinspringen in einen FOR- loop nicht definiert u. a. Daher wurde auf die Zulassung dieser Sprung-Kommandos verzichtet. Der einzige erlaubte Sprungbefehl ist die STOP-Anweisung, die aus jeder Struktur zum Programmende verzweigt.

"User Files Not Closed"
"Open Data Files: before: 0x0000, after: 0x0010"

Es wurden vom Benutzer Datenfiles geöffnet und nicht vor MACRO-Programmende geschlossen. Das System schließt diese Files und zeigt in hexadezimaler Bit-Form an, welche Files (Handle-Nummer) vor Beginn des Macro-Programmes geöffnet waren und welche vor Beendigung. Im obigen Beispiel war vorher kein File geöffnet, nachher der File mit dem Handle 4 (=0x0010, jedes Bit ist hier ein File-Handle)

"Too many DEFINT's"
"Too many DEFFLT's"
"Error: Max. array index [0..n], plus n named vars, exiting..."
"Warning: Max. array index [0..n], plus n named vars"
"Error: Max. string index [n], exiting..."
"Warning: Max. string index [n]"
Es sind bis zu 256 Integer und 256 Float Variable mit Namen zugelassen. Dazu gibt es 256 Integer-Array-Variable und 256 Float-Array-Variable und außerdem 16 String-Variable. Wenn die Überschreitung innerhalb einer Parameterzeile auftritt, dann gibt es nur eine Warnung, sonst, d. h. bei einer Variablenzuweisung, einen Fehlerabbruch.

"DEFINT: Double Define"
"DEFFLT: Double Define"
Variable wurde doppelt definiert. Achtung: Variablennamen werden z. Zt. nur bis inclusive dem 12. Zeichen überprueft und verglichen.

"Not a valid string"
Ein String muß immer in Hochkommata ("-Zeichen) gesetzt werden.