Crimson® 3.1

Referenzhandbuch | July 2020

LP1045 | Revision E

COPYRIGHT

SOFTWARE LICENSE

Software supplied with each Red Lion® product remains the exclusive property of Red Lion. Red Lion grants with each unit a perpetual license to use this software with the express limitations that the software may not be copied or used in any other product for any purpose. It may not be reverse engineered, or used for any other purpose other than in and with the computer hardware sold by Red Lion.

Red Lion Controls, Inc.
20 Willow Springs Circle
York, PA 17406

KONTAKTINFORMATIONEN:

AMERIKAS

In den USA: +1 (877) 432-9908
Außerhalb der USA: +1 (717) 767-6511
Stunden: 8 am-6 pm Östliche Standardzeit
(UTC/GMT -5 stunden)

ASIEN-PAZIFIK

Shanghai, V.R. China: +86 21-6113-3688 x767
Stunden: 9 am-6 pm China Standardzeit
(UTC/GMT +8 stunden)

EUROPA

Niederlande: +31 33-4723-225
Frankreich: +33 (0) 1 84 88 75 25
Deutschland: +49 (0) 1 89 5795-9421
GROßBRITANNIEN: +44 (0) 20 3868 0909
Stunden: 9 am-5 pm Mitteleuropäische Zeit
(UTC/GMT +1 stunde)

Webseite: www.redlion.net
Unterstützung: support.redlion.net

Inhaltsverzeichnis

Vorwort

Disclaimer

Obwohl alle Anstrengungen unternommen wurden, um sicherzustellen, dass dieses Dokument zum Zeitpunkt der Veröffentlichung vollständig und exakt ist, sind Änderungen an den darin enthaltenen Informationen vorbehalten. Red Lion Controls, Inc. ist nicht verantwortlich für Ergänzungen oder Änderungen des Originaldokuments. Industrielle Netzwerke variieren stark in der Konfiguration, Topologie und den Verkehrsbedingungen. Dieses Dokument ist nur als allgemeine Anleitung gedacht. Es wurde nicht auf alle möglichen Anwendungen getestet und es kann in manchen Situationen nicht vollständig oder genau sein.

Dieses Handbuch ist für Mitarbeiter gedacht, die für die Konfiguration und Inbetriebnahme von Crimson-Geräten für die Verwendung in Visualisierungs-, Überwachungs- und Steuerungsanwendungen verantwortlich sind. Benutzer dieses Dokuments sind aufgefordert, die Warnungen und Vorsichtsmaßnahmen im gesamten Dokument zu beachten.

Markenhinweise

Red Lion Controls, Inc. erkennt das Eigentum der folgenden in diesem Dokument verwendeten geschützten Begriffe an.

- EtherNet/IP™und CIP™ sind Marken von ODVA.
- Microsoft®, Windows®, Windows NT®, und Windows Vista™ sind entweder eingetragene Marken oder Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

Alle anderen Firmen- und Produktnamen sind Marken der jeweiligen Eigentümer.

Dokumentverlauf und zugehörige Veröffentlichungen

Die Versionen dieses Dokuments auf Papier und elektronischen Medien werden nur bei der Veröffentlichung von Hauptversionen revidiert und enthalten deshalb möglicherweise nicht die neuesten Produktinformationen. Gegebenenfalls werden Technische Hinweise und/oder Produktzusätze mit den neuen Informationen oder Dokumentänderungen zwischen Hauptversionen zur Verfügung gestellt.

Die neueste Online-Version dieses Dokuments kann über die Red Lion-Website unter https://www.redlion.net/red-lion-software/crimson/crimson-31 aufgerufen werden.

Zusätzliche Produktinformationen

Zusätzliche Produktinformationen können von Ihrem lokalen Außendienstmitarbeiter oder von Red Lion über die auf der Innenseite der Vorderabdeckung aufgelisteten Kontaktnummern und/oder E-Mail-Adressen angefordert werden.

Kapitel 1 Einführung

Crimson® 3.1 ist die neueste Version der weithin anerkannten Crimson-Gerätekonfigurationssoftware von Red Lion. Dieses Referenzhandbuch ist eine Ergänzung zum Crimson 3.1 User Manual mit Details zu den Standardfunktionen und Systemvariablen, die in Crimson 3.1 verfügbar sind. Diese Funktionen helfen Crimson 3.1-Benutzern, noch einfacher und effizienter leistungsstarke und attraktive Crimson-Gerätelösungen zu entwickeln.

Unterstützte Geräte

Crimson 3.1 unterstützt nur die Red Lion-Produkte, die über die erforderliche Speicherkapazität und die Prozessorleistung verfügen, um die zusätzlichen Funktionen der Software zu implementieren. Das bedeutet, dass die Graphite-Produktfamilie der Mensch-Maschine-Schnittstellen (HMIs) und Steuerungen mit Crimson 3.1 konfiguriert werden können; die Produktfamilien G3 HMI und G3 Kadet werden nicht unterstützt. Wir gehen davon aus, dass Sie Ihre G3 HMI- und Kadet-Anwendungen zu der neuen CR3000- bzw. die CR1000-Serie migrieren.

Darüber hinaus werden die derzeit verfügbaren Versionen unserer Produkte Data Station Plus, Modular Controller und ProducTVity Station ebenfalls nicht von Crimson 3.1 unterstützt. Konfigurieren Sie diese Geräte mit Crimson 3.0, bis aktualisierte Versionen verfügbar sind.

Systemanforderungen

Crimson 3.1 ist für die Ausführung unter einer beliebigen Version von Microsoft Windows ab Windows 7 ausgelegt. Die Speicheranforderungen sind gering, und auf allen Systemen, die die Mindestsystemanforderungen für das Betriebssystem erfüllen, kann Crimson 3.1 ausgeführt werden. Für die Installation sind etwa 600 MB freier Speicherplatz erforderlich, und Sie sollten idealerweise einen Bildschirm mit ausreichender Auflösung haben, um die Bearbeitung von Bildschirmseiten ohne Scrollen zu ermöglichen.

Überprüfen auf Updates

Wenn Sie über eine Internetverbindung verfügen, können Sie mit dem Befehl “Auf Update überprüfen” im Menü “Hilfe” auf der Red Lion-Website nach einer neuen Version von Crimson 3.1 suchen. Wenn eine neuere Version als die von Ihnen verwendete Version gefunden wird, werden Sie von Crimson gefragt, ob Sie das Upgrade herunterladen und die Software automatisch aktualisieren möchten. Sie können das Upgrade auch manuell von der Red Lion-Website herunterladen, indem Sie die Seite “Downloads” im Bereich “Support” besuchen.

Hilfe und Unterstützung

Wenn Sie auf ein Problem stoßen oder Hilfe benötigen, sind die folgenden Ressourcen verfügbar.

Technischer Support

Technischen Support finden Sie im Internet unter: support.redlion.net
You may also call:
In den USA: +1 (877) 432-9908
Außerhalb der USA: +1 (717) 767-6511

Online-Foren

Es gibt eine Reihe von Online-Foren zur Unterstützung von Benutzern von SPS und HMIs. Red Lion empfiehlt das Q&A-Forum unter http://www.plctalk.net/qanda/. Das Diskussionsforum ist mit vielen hilfsbereiten Experten bevölkert, und die technischen Support-Mitarbeiter von Red Lion überwachen dieses Forum auf Fragen zu unseren Produkten.

Kapitel 2 Standardfunktionen

In diesem Kapitel werden die Standardfunktionen beschrieben, die in Crimson 3.1 zur Verfügung gestellt werden. Diese Funktionen können innerhalb von Programmen, Aktionen oder Ausdrücken aufgerufen werden, wie im Crimson 3.1 User Manual beschrieben. Funktionen, die als *aktiv* gekennzeichnet sind, dürfen nicht in Ausdrücken verwendet werden, die keine Werte ändern dürfen, wie z. B. im steuernden Ausdruck eines Anzeigeprimitivs. Funktionen, die als *passiv* gekennzeichnet sind, können in jedem Kontext verwendet werden.

Abs(*value*)

ARGUMENT	TYP	BESCHREIBUNG
`value`	`int / float`	Der zu verarbeitende Wert.

Beschreibung

Gibt den absoluten Wert des Arguments zurück. Mit anderen Worten: Wenn value ein positiver Wert ist, wird dieser Wert zurückgegeben; wenn value negativ ist, wird ein Wert der gleichen Größe, jedoch mit umgekehrtem Vorzeichen zurückgegeben.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int` oder `float`, abhängig vom Typ des Arguments für `value`.

Beispiel

```
Error = abs(PV - SP)
```

AbsR64(result, tag)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag`	`int`	Das Tag, für das der absolute Wert berechnet werden soll.

Beschreibung

Berechnet den absoluten Wert von *tag* mithilfe von 64-Bit-Gleitkommazahlen (doppelte Genauigkeit) und speichert das Ergebnis in *result*. Der Eingabeoperand *tag* sollte von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu AddR64 ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
void
```

Beispiel

```
AbsR64(result[0], tag[0])
```

acos(*value*)

ARGUMENT	TYP	BESCHREIBUNG
`value`	`float`	Der zu verarbeitende Wert.

Beschreibung

Gibt den Winkel `theta` in Radianten zurück, sodass `cos(theta)` gleich `value` ist.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
float
```

Beispiel

```
theta = acos(1.0)
```

acosR64(result, tag)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag`	`int`	Der zu verarbeitende Wert

Beschreibung

Berechnet den Arcuscosinus von *tag* unter Verwendung von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in `result`. Der Eingabeoperand `tag` sollte von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu AddR64 ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
void
```

Beispiel

```
acosR64(result[0], tag[0])
```

AddR64(result, tag1, tag2)

ARGUMENT	TYP	BESCHREIBUNG
result	int	Das Ergebnis.
tag1	int	Das erste Summanden-Tag.
tag2	int	Das zweite Summanden-Tag.

Beschreibung

Berechnet den Wert von tag1 plus tag2 mithilfe von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in result. Die Eingabeoperanden tag1 und tag2 sollten von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Der Wert von result kann für weitere 64-Bit-Berechnungen verwendet oder mit der Funktion AsTextR64 für die Anzeige als Zeichenfolge formatiert werden.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

void

Beispiel

Dieses Beispiel zeigt die Berechnung von π + 2 mit 64-Bit-Mathematik.
Operand1, Operand2 und Result sind ganzzahlige Array-Tags, jeweils mit einer Länge von 2.

```
int NumberTwo = 2;

cstring PiString = "3.14159265358979";

IntToR64(Operand1[0], NumberTwo);

TextToR64(PiString, Operand2[0]);

AddR64(Result[0], Operand1[0], Operand2[0]);

cstring PiPlusTwo = AsTextR64(Result[0]);
```

PiPlusTwo enthält jetzt “5.141592654”, also “π + 2” als Zeichenfolge dargestellt.

AddU32 (tag1, tag2)

ARGUMENT	TYP	BESCHREIBUNG
`tag1`	`int`	Das erste Summanden-Tag.
`tag2`	`int`	Das zweite Summanden-Tag.

Beschreibung

Gibt den Wert von *tag1* plus *tag2* in einem Kontext ohne Vorzeichen zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiel

```
Result = AddU32(tag1, tag2)
```

AlarmAccept(*alarm*)

ARGUMENT	TYP	BESCHREIBUNG
`alarm`	`int`	Ein Wert, der den Alarm kodiert, der akzeptiert werden soll.

Beschreibung

Diese Funktion ist in der aktuellen Version **nicht** implementiert.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

AlarmAcceptAll()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Akzeptiert alle aktiven Alarme.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
AlarmAcceptAll()
```

AlarmAcceptEx(*source*, *method*, *code*)

ARGUMENT	TYP	BESCHREIBUNG
`source`	`int`	Die Quelle des Alarms.
`method`	`int`	Die Akzeptanzmethode.
`code`	`int`	Der Akzeptanzcode.

Beschreibung

Nimmt einen Alarm an, der von einem Rich-Communication-Treiber gemeldet wurde, der selbst in der Lage ist, Alarme und Ereignisse zu generieren. Diese Funktion wird nicht von Treibern verwendet, die derzeit im Lieferumfang von Crimson 3.1 enthalten sind.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

AlarmAcceptTag(*tag, index, event*)

ARGUMENT	TYP	BESCHREIBUNG
`tag`	`int`	Der Index des Tags, für den der Alarm definiert ist.
`index`	`int`	Das relevante Element eines Array-Tags, andernfalls Null.
`event`	`int`	Entweder 1 oder 2, je nachdem, welcher Alarm angenommen wird.

Beschreibung

Nimmt einen von einem Tag generierten Alarm an. Die Argumente geben die Tag-Nummer und die Alarmnummer an und können optional ein Array-Element enthalten. Wenn Sie einen Alarm von Tags akzeptieren, die keine Arrays sind, setzen Sie die Elementnummer auf Null.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
AlarmAcceptTag(10, 0, 1)
```

asin(*value*)

ARGUMENT	TYP	BESCHREIBUNG
`value`	`float`	Der zu verarbeitende Wert.

Beschreibung

Gibt den Winkel `theta` in Radianten zurück, sodass `sin(theta)` gleich `value` ist.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`float`

Beispiel

```
theta = asin(1.0)
```

asinR64(result, tag)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag`	`int`	Der zu verarbeitende Wert.

Beschreibung

Berechnet den Arcussinus von *tag* unter Verwendung von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in *result*. Der Eingabeoperand *tag* sollte von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu AddR64 ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
void
```

Beispiel

```
asinR64(result[0], tag[0])
```

AsText(*n*)

ARGUMENT	TYP	BESCHREIBUNG
`n`	`int / float`	Der Wert, der in Text konvertiert werden soll.

Beschreibung

Gibt den numerischen Wert als Zeichenfolge formatiert zurück. Die durchgeführte Formatierung entspricht der, die beim allgemeinen numerischen Format durchgeführt wird. Beachten Sie, dass numerische Tags in Zeichenfolgen konvertiert werden können, indem sie ihre `AsText`-Eigenschaft verwenden und beispielsweise auf `Tag1.AsText` verweisen.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
cstring
```

Beispiel

```
Text = AsText(Tag1 / Tag2)
```

AsTextR64(data)

ARGUMENT	TYP	BESCHREIBUNG
`data`	`int`	Der 64-Bit-Gleitkommawert, der konvertiert werden soll.

Beschreibung

Konvertiert den in *data* gespeicherten Wert von einem 64-Bit-Gleitkommawert in eine Zeichenfolge, die für die Anzeige geeignet ist. Das Tag *data* muss ein ganzzahliges Array mit einer Länge von mindestens 2 sein. Der Wert von *data* wird in der Regel von einer der bereitgestellten 64-Bit-Gleitkomma-Mathematikfunktionen abgerufen. Ein Beispiel für die beabsichtigte Verwendung dieser Funktion finden Sie im Eintrag zu AddR64.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
cstring
```

Beispiel

```
Result = AsTextR64(data[0])
```

AsTextR64WithFormat(format, data)

ARGUMENT	TYP	BESCHREIBUNG
`format`	`cstring`	Eine Zeichenfolge, die die gewünschte Breite, Genauigkeit und Markierungen enthält.
`data`	`int`	Der 64-Bit-Gleitkommawert, der konvertiert werden soll.

Beschreibung

Konvertiert den in *data* gespeicherten Wert von einem 64-Bit-Gleitkommawert gemäß angegebenem *format* in eine Zeichenfolge, die für die Anzeige geeignet ist. Das *format* sollte als “width.precision.flags” (Breite.Genauigkeit.Markierungen) codiert werden, wobei die Breite die maximale Zeichenanzahl für den numerischen Teil der resultierenden Zeichenfolge definiert und die Genauigkeit die Anzahl von Ziffern rechts neben dem Dezimalzeichen in der resultierenden Zeichenfolge angibt. Die verfügbaren Markierungen sind 1, um führende Nullen anzuzeigen, und 2 zum Ausblenden einer nachgestellten 0. Das *Tag* data muss ein ganzzahliges Array mit einer Länge von mindestens 2 sein. Der Wert von *data* wird in der Regel von einer der bereitgestellten 64-Bit-Gleitkomma-Mathematikfunktionen abgerufen. Ein Beispiel für die beabsichtigte Verwendung dieser Funktion finden Sie im Eintrag zu AddR64.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
cstring
```

Beispiel

```
Result = AsTextR64WithFormat(“17.8.3”, data[0])
```

atan(*value*)

ARGUMENT	TYP	BESCHREIBUNG
`value`	`float`	Der zu verarbeitende Wert.

Beschreibung

Gibt den Winkel `theta` in Radianten zurück, sodass `tan(theta)` gleich `value` ist.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
float
```

Beispiel

```
theta = atan(1.0)
```

atan2(*a*, *b*)

ARGUMENT	TYP	BESCHREIBUNG
`a`	`float`	Der Wert der Seite, die sich gegenüber dem Winkel Theta befindet.
`b`	`float`	Der Wert der Seite, die an den Winkel Theta angrenzt.

Beschreibung

Diese Funktion entspricht `atan(a/b)`, mit der Ausnahme, dass auch die Vorzeichen von `a` und `b` berücksichtigt werden, sodass der Rückgabewert im entsprechenden Quadranten liegt. Sie ist außerdem in der Lage, einen Nullwert für `b` zu verarbeiten und so die Unendlichkeit zu vermeiden, die sich ergeben würde, wenn stattdessen das Einzelargument von `tan` verwendet wird.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
float
```

Beispiel

```
theta = atan2(1,1)
```

atanR64(result, tag)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag`	`int`	Der zu verarbeitende Wert.

Beschreibung

Berechnet den Arcustangens von *tag* unter Verwendung von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in *result*. Der Eingabeoperand *tag* sollte von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu AddR64 ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
void
```

Beispiel

```
atanR64(result[0], tag[0])
```

atan2R64(result, a, b)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`a`	`int`	Der Wert der Seite, die sich gegenüber dem Winkel Theta befindet.
`b`	`int`	Der Wert der Seite, die an den Winkel Theta angrenzt.

Beschreibung

Das Äquivalent von *atan(a/b)* unter Verwendung von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit. Das Ergebnis wird in *result* gespeichert. Diese Funktion berücksichtigt die Vorzeichen von *a* und *b* zur Berechnung des Werts für den entsprechenden Quadranten. Dies entspricht der atan2-Funktion mit doppelter Genauigkeit. Die Eingabeoperanden *a* und *b* sollten von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu AddR64 ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`void`

Beispiel

```
Atan2R64(result[0], a[0], b[0])
```

Beep(*freq, period*)

ARGUMENT	TYP	BESCHREIBUNG
`freq`	`int`	Die erforderliche Frequenz in Halbtönen.
`period`	`int`	Die erforderliche Zeitdauer in Millisekunden.

Beschreibung

Gibt den Crimson-Gerätepieper für die angegebene Zeitdauer in der angegebenen Tonlage aus. Wenn Sie einen Wert von Null für `period` übergeben, wird der Pieper ausgeschaltet. Signaltonanforderungen werden nicht in die Warteschlange gestellt, sodass der Aufruf der Funktion vorherige Aufrufe sofort außer Kraft setzt. Für die musikalischen Talente unter Ihnen – das Argument `freq` wird in Halbtönen kalibriert. Und ganz im Ernst: Die Signalfunktion kann eine nützliche Debugging-Hilfe sein, da sie eine asynchrone Methode der Signalgebung für die Handhabung eines Ereignisses oder die Ausführung eines Programmschritts bietet.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
Beep(60, 100)
```

CanGotoNext()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Gibt den Wert “true” oder “false” zurück, der angibt, ob ein Aufruf von `GotoNext()` eine Seitenänderung erzeugt. Der Wert “false” gibt an, dass im Seitenverlaufspuffer keine Seiten mehr vorhanden sind.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

CanGotoPrevious()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Gibt den Wert “true” oder “false” zurück, der angibt, ob ein Aufruf von `GotoPrevious()` eine Seitenänderung erzeugt. Der Wert “false” gibt an, dass im Seitenverlaufspuffer keine Seiten mehr vorhanden sind.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

ClearEvents()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Löscht die Liste der im Ereignisprotokoll angezeigten Ereignisse.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
ClearEvents()
```

CloseFile(*file*)

ARGUMENT	TYP	BESCHREIBUNG
`file`	`int`	Das Datei-Handle, das von `OpenFile` zurückgegeben wird.

Beschreibung

Schließt eine Datei, die zuvor in einem Aufruf von `FileOpen()` geöffnet wurde.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
CloseFile(hFile)
```

ColBlend(*data, min, max, col1, col2*)

ARGUMENT	TYP	BESCHREIBUNG
`data`	`float`	Der Datenwert, der zur Steuerung der Rechenoperation verwendet werden soll.
`min`	`float`	Der Mindestwert für *data*.
`max`	`float`	Der Höchstwert für *data*.
`col1`	`int`	Die erste Farbe, die ausgewählt wird, wenn *data* gleich *min* ist.
`col2`	`int`	Die zweite Farbe, die ausgewählt wird, wenn *data* gleich *max* ist.

Beschreibung

Gibt eine Farbe zurück, die durch Mischen von zwei anderen Farben erzeugt wird, wobei der Anteil der einzelnen Farben auf dem Wert von *data* im Verhältnis zu den durch *min* und *max* angegebenen Grenzwerten basiert. Diese Funktion ist nützlich, wenn Sie die Anzeigeprimitive animieren, indem Sie deren Farben ändern.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

ColFlash(*freq*, *col1*, *col2*)

ARGUMENT	TYP	BESCHREIBUNG
`freq`	`int`	Die Anzahl der Wechsel pro Sekunde.
`col1`	`int`	Die erste Farbe.
`col2`	`int`	Die zweite Farbe.

Beschreibung

Gibt eine alternierende Farbe zurück, die aus *col1* und *col2* ausgewählt wurde und den Zyklus *freq* Mal pro Sekunde durchläuft. Diese Funktion ist nützlich, wenn Sie die Anzeigeprimitive animieren, indem Sie deren Farben ändern.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

ColGetBlue(col)

ARGUMENT	TYP	BESCHREIBUNG
`col`	`int`	Die Farbe, aus der die Komponente ausgewählt werden soll.

Beschreibung

Gibt die blaue Komponente des angegebenen Farbwerts zurück. Die Komponente wird so skaliert, dass sie im Bereich zwischen 0 und 255 liegt, obwohl Crimson intern mit 5-Bit-Farbkomponenten arbeitet.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

ColGetGreen(col)

ARGUMENT	TYP	BESCHREIBUNG
`col`	`int`	Die Farbe, aus der die Komponente ausgewählt werden soll.

Beschreibung

Gibt die grüne Komponente des angegebenen Farbwerts zurück. Die Komponente wird so skaliert, dass sie im Bereich zwischen 0 und 255 liegt, obwohl Crimson intern mit 5-Bit-Farbkomponenten arbeitet.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

ColGetRed(col)

ARGUMENT	TYP	BESCHREIBUNG
`col`	`int`	Die Farbe, aus der die Komponente ausgewählt werden soll.

Beschreibung

Gibt die rote Komponente des angegebenen Farbwerts zurück. Die Komponente wird so skaliert, dass sie im Bereich zwischen 0 und 255 liegt, obwohl Crimson intern mit 5-Bit-Farbkomponenten arbeitet.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

ColGetRGB(r,g,b)

ARGUMENT	TYP	BESCHREIBUNG
`r`	`int`	Die rote Komponente.
`g`	`int`	Die grüne Komponente.
`b`	`int`	Die blaue Komponente.

Beschreibung

Gibt einen Farbwert zurück, der aus den angegebenen Komponenten erstellt wurde. Die Komponenten müssen im Bereich von 0 bis 255 liegen, obwohl Crimson intern mit 5-Bit-Farbkomponenten arbeitet.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

ColPick2(*pick, col1, col2*)

ARGUMENT	TYP	BESCHREIBUNG
`pick`	`int`	Die Bedingung, die zur Auswahl der Farbe verwendet werden soll.
`col1`	`int`	Die erste Farbe, die ausgewählt wird, wenn *pick* zutrifft (“true”).
`col2`	`int`	Die zweite Farbe, die ausgewählt wird, wenn *pick* nicht zutrifft (“false”).

Beschreibung

Gibt je nach Status von *pick* eine der angegebenen Farben zurück.
Entsprechende Ergebnisse können auch mit dem Auswahloperator `?:` erreicht werden.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

ColPick4(*data1*, *data2*, *col1*, *col2*, *col3*, *col4*)

ARGUMENT	TYP	BESCHREIBUNG
`data1`	`int`	Der erste Datenwert.
`data2`	`int`	Der zweite Datenwert.
`col1`	`int`	Der Wert, wenn sowohl Data1 als auch Data2 *true* sind.
`col2`	`int`	Der Wert, wenn Data1 *false* und Data2 *true* ist.
`col3`	`int`	Der Wert, wenn Data1 true und Data2 *false* ist.
`col4`	`int`	Der Wert, wenn sowohl Data1 als auch Data2 *false* sind.

Beschreibung

Gibt einen von vier Werten zurück, basierend auf dem Status *true* oder *false* von zwei Datenelementen.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

ColSelFlash(*enable, freq, col1, col2, col3*)

ARGUMENT	TYP	BESCHREIBUNG
`enable`	`int`	Ein Wert, der *true* sein muss, um Blinken zu ermöglichen.
`freq`	`int`	Die Häufigkeit, mit der das Blinken auftreten soll.
`col1`	`int`	Der Wert, der bei deaktiviertem Blinken zurückgegeben werden soll.
`col2`	`int`	Die erste blinkende Farbe.
`col3`	`int`	Die zweite blinkende Farbe.

Beschreibung

Wenn enable gleich *true* ist, wird eine alternierende Farbe zurückgegeben, die aus *col2* und *col3* ausgewählt wurde und den Zyklus *freq* Mal pro Sekunde durchläuft. Wenn *enable* gleich *false* ist, wird konstant *col1* zurückgegeben. Diese Funktion ist nützlich, wenn Sie die Anzeigeprimitive animieren, indem Sie deren Farben ändern.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

CommitAndReset()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Mit dieser Funktion werden alle retentiven Tags in den internen Flash-Speicher geschrieben und dann das Gerät zurückgesetzt. Sie ist für die Verwendung in Verbindung mit Funktionen vorgesehen, die Konfiguration des Geräts ändern und dann eine Rücksetzung erfordern, damit die Änderungen wirksam werden.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
CommitAndReset()
```

CompactFlashEject()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Stellt jeglichen Zugriff auf die Speicherkarte ein, wodurch ein sicheres Entfernen der Karte möglich ist.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
CompactFlashEject()
```

CompactFlashStatus()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Gibt den Status des Speicherkartensteckplatzes als Ganzzahl zurück.

WERT	STATUS	BESCHREIBUNG
0	Leer	Entweder ist keine Karte installiert oder die Karte wurde über einen Aufruf an die Funktion `CompactFlashEject` ausgeworfen.
1	Ungültig	Die Karte ist beschädigt, falsch formatiert oder überhaupt nicht formatiert. Denken Sie daran, dass nur FAT16 unterstützt wird.
2	Wird geprüft	Die HMI prüft den Status der Karte. Dieser Status tritt ein, wenn eine Karte erstmals in die HMI eingesetzt wird.
3	Wird formatiert	Die HMI formatiert die Karte. Dieser Status tritt ein, wenn vom programmierenden PC ein Formatierungsvorgang angefordert wird.
4	Gesperrt	Entweder schreibt das Crimson-Gerät auf die Karte, oder die Karte ist angemeldet und Windows greift darauf zu.
5	Angemeldet	Eine gültige Karte ist installiert, aber sie ist weder durch das Crimson-Gerät noch durch Windows gesperrt.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
d = CompactFlashStatus()
```

CompU32(tag1, tag2)

ARGUMENT	TYP	BESCHREIBUNG
`tag1`	`int`	Das zu vergleichende Tag.
`tag2`	`int`	Das Tag, mit dem verglichen wird.

Beschreibung

Vergleicht *tag1* mit *tag2* in einem Kontext ohne Vorzeichen. Gibt einen der folgenden Werte zurück:

-1	tag1 ist kleiner als tag2.
0	tag1 ist gleich tag2.
+1	tag1 ist größer als tag2.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiel

```
Result = CompU32(tag1, tag2)
```

ControlDevice(*device, enable*)

ARGUMENT	TYP	BESCHREIBUNG
`device`	`int`	Das zu aktivierende oder zu deaktivierende Gerät.
`enable`	`int`	Gibt an, ob das Gerät aktiviert oder deaktiviert ist.

Beschreibung

Ermöglicht der Datenbank, ein angegebenes Kommunikationsgerät zu deaktivieren oder zu aktivieren. Die Zahl, die zur Identifizierung des Geräts im Argument `device` platziert werden soll, kann in der Statusleiste der Kategorie “Kommunikationen” angezeigt werden, wenn der Gerätename markiert ist.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
ControlDevice(1, true)
```

Copy(*dest, src, count*)

ARGUMENT	TYP	BESCHREIBUNG
`dest`	`int / float`	Das erste Array-Element, in das kopiert werden soll.
`src`	`int / float`	Das erste Array-Element, aus dem kopiert werden soll.
`count`	`int`	Die Anzahl der zu verarbeitenden Elemente.

Beschreibung

Kopiert `count` (Anzahl) Array-Elemente aus `src` in `dest`.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
Copy(Save[0], Work[0], 100)
```

CopyFiles(*source, target, flags*)

ARGUMENT	TYP	BESCHREIBUNG
`source`	`cstring`	Der Pfad, aus dem die Dateien kopiert werden sollen.
`target`	`cstring`	Der Pfad, in den die Dateien kopiert werden sollen.
`flags`	`int`	Die Markierungen, die den Kopiervorgang steuern.

Beschreibung

Kopiert alle Dateien aus dem Verzeichnis *source* in das Verzeichnis *target*.
Mit den verschiedenen Bits in `flags` wird der Kopiervorgang wie folgt geändert:

BIT	GEWICHT	BESCHREIBUNG
0	1	Wenn diese Option festgelegt ist, wird der Vorgang rekursiv in Unterverzeichnissen durchgeführt.
1	2	Wenn festgelegt, werden vorhandene Dateien überschrieben. Wenn leer, bleiben die vorhandenen Dateien unverändert.
2	4	Wenn festgelegt, werden alle Dateien kopiert. Wenn leer, werden nur Dateien kopiert, die nicht am Ziel vorhanden sind oder die in der Quelle neuere Zeitstempel haben.

Der Rückgabewert der Funktion ist bei Erfolg *true* und bei einem Fehler *false*.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

Beispiel

```
CopyFiles("C:\LOGS", "C:\BACKUP", 1)
```

cos(*theta*)

ARGUMENT	TYP	BESCHREIBUNG
`theta`	`float`	Der zu verarbeitende Winkel, in Radianten.

Beschreibung

Gibt den Cosinus des Winkels `theta` zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`float`

Beispiel

```
xp = radius*cos(theta)
```

cosR64(result, tag)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag`	`int`	Der zu verarbeitende Winkel, in Radianten.

Beschreibung

Berechnet den Cosinus von *tag* unter Verwendung von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in *result*. Der Eingabeoperand *tag* sollte von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu AddR64 ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
void
```

Beispiel

```
cosR64(result[0], tag[0])
```

CreateDirectory(*name*)

ARGUMENT	TYP	BESCHREIBUNG
`name`	`cstring`	Das Verzeichnis, das erstellt werden soll.

Beschreibung

Erstellt ein neues Verzeichnis auf der Speicherkarte. Das Ablagesystem von Crimson 3.1 unterstützt jetzt sowohl FAT16 als auch FAT32. Wenn die Speicherkarte mit FAT32 formatiert wurde, können lange Dateinamen verwendet werden. Wenn die Speicherkarte mit FAT16 formatiert wurde, werden lange Dateinamen nicht unterstützt. Beachten Sie, dass bei Verwendung von umgekehrten Schrägstrichen im Pfadnamen zur Trennung von Pfadelementen diese gemäß den Crimson-Regeln für Zeichenfolgenkonstanten verdoppelt werden müssen, wie im Crimson 3.1 User Manual im Kapitel zum Schreiben von Ausdrücken beschrieben. Um diese Komplikation zu vermeiden, können Schrägstriche anstelle von umgekehrten Schrägstrichen verwendet werden. Schrägstriche müssen nicht verdoppelt werden. Die Funktion gibt bei Erfolg einen Wert von 1 zurück, bei einem Fehler den Wert 0.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
int
```

Beispiel

```
Result = CreateDirectory("/LOGS/LOG1")
```

CreateFile(*name*)

ARGUMENT	TYP	BESCHREIBUNG
`name`	`cstring`	Die zu erstellende Datei.

Beschreibung

Erstellt eine leere Datei auf der Speicherkarte. Das Ablagesystem von Crimson 3.1 unterstützt jetzt sowohl FAT16 als auch FAT32. Wenn die Speicherkarte mit FAT32 formatiert wurde, können lange Dateinamen verwendet werden. Wenn die Speicherkarte mit FAT16 formatiert wurde, werden lange Dateinamen nicht unterstützt. Beachten Sie, dass bei Verwendung von umgekehrten Schrägstrichen im Pfadnamen zur Trennung von Pfadelementen diese gemäß den Crimson-Regeln für Zeichenfolgenkonstanten verdoppelt werden müssen, wie im Crimson 3.1 User Manual im Kapitel zum Schreiben von Ausdrücken beschrieben. Um diese Komplikation zu vermeiden, können Schrägstriche anstelle von umgekehrten Schrägstrichen verwendet werden. Schrägstriche müssen nicht verdoppelt werden. Die Funktion gibt bei Erfolg einen Wert von 1 zurück, bei einem Fehler den Wert 0. Beachten Sie, dass die Datei nicht geöffnet wird, nachdem sie erstellt wurde – es muss ein nachfolgender Aufruf an `OpenFile()` erfolgen, damit Daten gelesen oder geschrieben werden können.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
int
```

Beispiel

```
Success = CreateFile("/logs/custom/myfile.txt")
```

DataToText(*data, limit*)

ARGUMENT	TYP	BESCHREIBUNG
`data`	`int`	Das erste Element in einem Array.
`limit`	`int`	Die Anzahl der zu verarbeitenden Zeichen.

Beschreibung

Bildet eine Zeichenfolge aus einem Array und extrahiert aus jedem numerischen Array-Element 4 Zeichen, bis entweder die Grenze erreicht ist oder ein Nullzeichen erkannt wird.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
cstring
```

Beispiel

```
string = DataToText(Data[0], 8)
```

Date(*y*, *m*, *d*)

ARGUMENT	TYP	BESCHREIBUNG
`y`	`int`	Das zu codierende Jahr in vierstelliger Form.
`m`	`int`	Der zu codierende Monat von 1 bis 12.
`d`	`int`	Das Datum, das codiert werden soll, ab 1 aufwärts.

Beschreibung

Gibt einen Wert zurück, der das angegebene Datum als Anzahl der Sekunden seit dem Bezugspunkt vom 1. Januar 1997 angibt. Dieser Wert kann dann mit anderen Zeit-/Datumsfunktionen verwendet werden.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
t = Date(2000,12,31)
```

DecR64(result, tag)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag`	`int`	Der zu verarbeitende Wert.

Beschreibung

Verringert den Wert von *tag* unter Verwendung von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit um Eins und speichert das Ergebnis in *result*. Dies entspricht dem Operator -- mit doppelter Genauigkeit. Der Eingabeoperand *tag* sollte von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu AddR64 ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
void
```

Beispiel

```
DecR64(result[0], tag[0])
```

DecToText(*data, signed, before, after, leading, group*)

ARGUMENT	TYP	BESCHREIBUNG
`data`	`int/float`	Die numerischen Daten, die formatiert werden sollen.
`signed`	`int`	0 – ohne Vorzeichen, 1 – weiches Vorzeichen, 2 – hartes Vorzeichen.
`before`	`int`	Die Anzahl der Ziffern links vom Dezimalzeichen.
`after`	`int`	Die Anzahl der Ziffern rechts vom Dezimalzeichen.
`leading`	`int`	0 – keine führenden Nullen, 1 – führende Nullen.
`group`	`int`	0 – keine Gruppierung, 1 – Dreiergruppierung von Zahlen.

Beschreibung

Formatiert den Wert in `data` als Dezimalwert für die restlichen Parameter. Die Funktion wird in der Regel verwendet, um erweiterte Formatierungsoptionen über Programme zu erstellen oder um Zeichenfolgen vorzubereiten, die über einen RAW-Anschlusstreiber gesendet werden.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
cstring
```

Beispiel

```
Text = DecToText(var1, 2, 5, 2, 0, 1)
```

Deg2Rad(*theta*)

ARGUMENT	TYP	BESCHREIBUNG
`theta`	`float`	Der zu verarbeitende Winkel.

Beschreibung

Gibt `theta` zurück, der von Grad in Radianten konvertiert wurde.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`float`

Beispiel

```
Load = Weight * cos(Deg2Rad(Angle))
```

DeleteDirectory(*name*)

ARGUMENT	TYP	BESCHREIBUNG
`name`	`cstring`	Das Verzeichnis, das gelöscht werden soll.

Beschreibung

Entfernt ein leeres Verzeichnis (d. h. ein Verzeichnis, das keine Dateien und/oder Unterverzeichnisse enthält) von der Speicherkarte. Das Ablagesystem von Crimson 3.1 unterstützt jetzt sowohl FAT16 als auch FAT32. Wenn die Speicherkarte mit FAT32 formatiert wurde, können lange Dateinamen verwendet werden. Wenn die Speicherkarte mit FAT16 formatiert wurde, werden lange Dateinamen nicht unterstützt. Beachten Sie, dass bei Verwendung von umgekehrten Schrägstrichen im Pfadnamen zur Trennung von Pfadelementen diese gemäß den Crimson-Regeln für Zeichenfolgenkonstanten verdoppelt werden müssen, wie im Crimson 3.1 User Manual im Kapitel zum Schreiben von Ausdrücken beschrieben. Um diese Komplikation zu vermeiden, können Schrägstriche anstelle von umgekehrten Schrägstrichen verwendet werden. Schrägstriche müssen nicht verdoppelt werden. Die Funktion gibt bei Erfolg einen Wert von 1 zurück, bei einem Fehler den Wert 0.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
int
```

Beispiel

```
Success = DeleteDirectory("/logs/custom")
```

DeleteFile(*file*)

ARGUMENT	TYP	BESCHREIBUNG
`file`	`int`	Das Datei-Handle, wie es von OpenFile zurückgegeben wurde.

Beschreibung

Schließt eine Datei auf der Speicherkarte und löscht sie anschließend davon. Die Datei muss zunächst in einem beschreibbaren Status geöffnet werden.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

Beispiel

hFile = OpenFile(“/LOGS/LOG1/01010101.csv”, 1)

```
Result = DeleteFile(hFile)
```

DevCtrl(*device, function, data*)

ARGUMENT	TYP	BESCHREIBUNG
`device`	`int`	Der Index des zu steuernden Geräts.
`function`	`int`	Die erforderliche Funktion, die ausgeführt werden soll.
`data`	`cstring`	Beliebiger Parameter für die Funktion.

Beschreibung

Diese Funktion wird verwendet, um einen speziellen Vorgang auf einem Kommunikationsgerät durchzuführen. Die Zahl, die zur Identifizierung des Geräts im Argument `device` platziert werden soll, kann in der Statusleiste der Kategorie “Kommunikationen” angezeigt werden, wenn der Gerätename markiert ist. Die durchzuführende Aktion wird durch den Parameter `function` angegeben, dessen Werte vom Typ des angesprochenen Geräts abhängen. Mit dem Parameter `data` können zusätzliche Informationen an den Treiber übergeben werden. Die meisten Treiber unterstützen diese Funktion nicht. Sofern unterstützt, sind die Vorgänge treiberspezifisch und sind separat dokumentiert.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

Beispiel

Spezifische Beispiele finden Sie in den Anwendungshinweisen für den Kommunikationstreiber.

DisableDevice(*device*)

ARGUMENT	TYP	BESCHREIBUNG
`device`	`int`	Das zu deaktivierende Gerät.

Beschreibung

Deaktiviert die Kommunikation für das angegebene Gerät. Die Zahl, die zur Identifizierung des Geräts im Argument `device` platziert werden soll, kann in der Statusleiste der Kategorie “Kommunikationen” angezeigt werden, wenn der Gerätename markiert ist.

Funktionstyp

Die Funktion ist *passiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
DisableDevice(1)
```

DispOff()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Schaltet die Hintergrundbeleuchtung des Displays aus.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
DispOff()
```

DispOn()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Schaltet die Hintergrundbeleuchtung des Displays ein.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
DispOn()
```

DivR64(result, tag1, tag2)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag1`	`int`	Der Dividend.
`tag2`	`int`	Der Divisor.

Beschreibung

Berechnet den Wert von *tag1* geteilt durch *tag2* mithilfe von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in *result*. Dies entspricht *tag1/tag2* mit doppelter Genauigkeit. Die Eingabeoperanden *tag1* und *tag2* sollten von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Unter AddR64 ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`void`

Beispiel

```
DivR64(result[0], tag1[0], tag2[0])
```

DivU32(tag1, tag2)

ARGUMENT	TYP	BESCHREIBUNG
`tag1`	`int`	Der Dividend.
`tag2`	`int`	Der Divisor.

Beschreibung

Gibt den Wert von *tag1* geteilt durch *tag2* in einem Kontext ohne Vorzeichen zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiel

```
Result = DivU32(tag1, tag2)
```

DrvCtrl(*port, function, data*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Der Index des zu steuernden Treibers.
`function`	`int`	Die erforderliche Funktion, die ausgeführt werden soll.
`data`	`int`	Beliebiger Parameter für die Funktion.

Beschreibung

Diese Funktion wird verwendet, um einen speziellen Vorgang an einem Kommunikationstreiber durchzuführen. Die Zahl, die im Argument `port` platziert werden soll, um den Treiber zu identifizieren, ist die Portnummer, an die der Treiber gebunden ist. Die durchzuführende Aktion wird durch den Parameter `function` angegeben, dessen Werte vom Treiber abhängen. Mit dem Parameter `data` können zusätzliche Informationen an den Treiber übergeben werden. Die meisten Treiber unterstützen diese Funktion nicht. Sofern unterstützt, sind die Vorgänge treiberspezifisch und sind separat dokumentiert.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

Beispiel

Spezifische Beispiele finden Sie in den Anwendungshinweisen für den Kommunikationstreiber, online erhältlich unter: http:// www.redlion.net/red-lion-software/crimson/crimson-31.

EjectDrive(*drive*)

ARGUMENT	TYP	BESCHREIBUNG
`drive`	`int`	Der Laufwerksbuchstabe des Laufwerks, das ausgeworfen werden soll.

Beschreibung

Wirft ein an das System angeschlossenes Wechsellaufwerk aus, sodass ein sicheres Entfernen des Geräts möglich ist.

Laufwerk C bezieht sich auf die Speicherkarte, während sich Laufwerk D auf den USB-Speicherstick bezieht.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
EjectDrive('C')
```

EmptyWriteQueue (*dev*)

ARGUMENT	TYP	BESCHREIBUNG
`dev`	`int`	Die Gerätenummer.

Beschreibung

Leert die Schreibwarteschlange für das durch das Argument `dev` identifizierte Gerät. Dadurch werden alle ausstehenden Schreibvorgänge für das Gerät aus der Warteschlange entfernt, und die entfernten Informationen werden nicht auf das Gerät übertragen. Die Gerätenummer kann in der Statusleiste von Crimson identifiziert werden, wenn in “Kommunikation” ein Gerät ausgewählt ist.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
EmptyWriteQueue(1)
```

EnableBatteryCheck(*disable*)

ARGUMENT	TYP	BESCHREIBUNG
`disable`	`int`	Setzen Sie diesen Wert auf 1, um die Überprüfung zu deaktivieren.

Beschreibung

Aktivieren oder deaktivieren Sie die Batterieprüfung, die nach dem Starten des Systems durchgeführt wird. Setzen Sie *disable* auf 0, um die Batterieprüfung zu aktivieren. Setzen Sie *disable* auf 1, um die Batterieprüfung zu deaktivieren. Wenn die Batterieprüfung aktiviert ist und der Ladezustand der Batterie niedrig ist, zeigt das System eine Warnung an, um den Benutzer zu informieren. Der Benutzer muss entweder 60 Sekunden warten oder die Anweisungen auf dem Bildschirm befolgen, um nach der Warnung fortzufahren. Wenn die Batterieprüfung deaktiviert ist, wird die Batteriewarnung nie angezeigt, unabhängig vom Ladezustand der Batterie.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
void
```

Beispiel

```
EnableBatteryCheck(0)
```

EnableDevice(*device*)

ARGUMENT	TYP	BESCHREIBUNG
`device`	`int`	Das zu aktivierende Gerät.

Beschreibung

Aktiviert die Kommunikation für das angegebene Gerät. Die Zahl, die zur Identifizierung des Geräts im Argument `device` platziert werden soll, kann in der Statusleiste der Kategorie “Kommunikationen” angezeigt werden, wenn der Gerätename markiert ist.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
EnableDevice(1)
```

EndBatch()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Stoppt den aktuellen Batch. Beachten Sie, dass das Starten eines neuen Batches innerhalb von weniger als 10 Sekunden nach Beendigung oder Start des letzten Batches zu nicht definiertem Verhalten führt. Um direkt von einem Batch zu einem anderen zu wechseln, rufen Sie `NewBatch()` ohne einen dazwischen liegenden Aufruf von `EndBatch()` auf.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
Result = EndBatch()
```

EndModal(*code*)

ARGUMENT	TYP	BESCHREIBUNG
`code`	`int`	Der Wert, der an den Aufrufer von ShowModal zurückgegeben wird

Beschreibung

Modale Popups, die mit der Funktion `ShowModal()` angezeigt werden sollen, werden sofort angezeigt. Die Funktion `ShowModal()` wird erst nach einer Aktion namens `EndModal()` auf der Popup-Seite zurückgegeben, um den von letzterer übergebenen Wert an den Aufrufer der vorherigen Funktion zu übergeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

EnumOptionCard(*s*)

ARGUMENT	TYP	BESCHREIBUNG
`s`	`int`	Der Optionskartensteckplatz, entweder 0 oder 1

Beschreibung

Gibt den Typ der Optionskarte zurück, der für den angegebenen Steckplatz konfiguriert ist.
Folgende Werte können zurückgegeben werden:

WERT	KARTENTYP
0	Keine
1	Seriell
2	CAN
3	Profibus
4	FireWire
5	DeviceNet
6	CAT Link
7	Modem
8	MPI
9	Ethernet
10	USB-Host

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

EqualR64(*a*, *b*)

ARGUMENT	TYP	BESCHREIBUNG
a	int	Der erste zu vergleichende Wert.
b	int	Der zweite zu vergleichende Wert.

Beschreibung

Vergleicht den Wert von *a* mit *b* mithilfe von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und gibt 1 zurück, wenn *a* gleich *b* ist, andernfalls 0. Dies entspricht a == b mit doppelter Genauigkeit. Der Vergleich von Gleitkommawerten für exakte Gleichheit ist aufgrund von Rundungsfehlern fehleranfällig. Die Eingabeoperanden *a* und *b* sollten von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu AddR64 ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
EqualR64(a[0], b[0])
```

exp(*value*)

ARGUMENT	TYP	BESCHREIBUNG
`value`	`float`	Der zu verarbeitende Wert.

Beschreibung

Gibt *e* (2,7183...) hoch `value` zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`float`

Beispiel

```
Variable2 = exp(1.609)
```

exp10(*value*)

ARGUMENT	TYP	BESCHREIBUNG
`value`	`float`	Der zu verarbeitende Wert.

Beschreibung

Gibt 10 hoch `value` zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`float`

Beispiel

```
Variable4 = exp10(0.699)
```

exp10R64(*result, tag*)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag`	`int`	Der zu verarbeitende Wert.

Beschreibung

Berechnet den Wert von 10 hoch *tag* unter Verwendung von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in *result*. Der Eingabeoperand *tag* sollte von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu AddR64 ist ein detailliertes Beispiel angegeben .

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`void`

Beispiel

```
Exp10R64(result[0], tag1[0])
```

expR64(*result, tag*)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag`	`int`	Der zu verarbeitende Wert.

Beschreibung

Berechnet den Wert von *e* (2,7183...) hoch *tag* unter Verwendung von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in *result*. Der Eingabeoperand *tag* sollte von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu AddR64 ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`void`

Beispiel

```
expR64(result[0], tag[0])
```

FileSeek(*file*, *pos*)

ARGUMENT	TYP	BESCHREIBUNG
`file`	`int`	Das Datei-Handle, das von `OpenFile` zurückgegeben wird.
`pos`	`int`	Die Position innerhalb der Datei.

Beschreibung

Verschiebt den Dateizeiger für die angegebene Datei an den angegebenen Speicherort.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

FileTell(*file*)

ARGUMENT	TYP	BESCHREIBUNG
`file`	`int`	Das Datei-Handle, das von `OpenFile` zurückgegeben wird.

Beschreibung

Gibt den aktuellen Wert des Dateizeigers für die angegebene Datei zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Fill(*element, data, count*)

ARGUMENT	TYP	BESCHREIBUNG
`element`	`int / float`	Das erste Array-Element, das verarbeitet werden soll.
`data`	`int / float`	Der zu schreibende Datenwert.
`count`	`int`	Die Anzahl der zu verarbeitenden Elemente.

Beschreibung

Legt `count` (Anzahl) Array-Elemente ab `element` auf gleich `data` fest.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
Fill(List[0], 0, 100)
```

Find(*string*, *char*, *skip*)

ARGUMENT	TYP	BESCHREIBUNG
`string`	`cstring`	Die zu bearbeitende Zeichenfolge.
`char`	`int`	Das zu suchende Zeichen.
`skip`	`int`	Gibt an, wie oft das Zeichen übersprungen wird.

Beschreibung

Gibt die Position von `char` in `string` zurück, wobei die ersten `skip` Vorkommen ignoriert werden. Die erste gezählte Position ist 0. Gibt -1 zurück, wenn `char` nicht gefunden wird. Im nachfolgenden Beispiel ist die Position des Punkts 7, wobei das erste Vorkommen übersprungen wird.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
Position = Find("one:two:three",':',1)
```

FindFileFirst(*dir*)

ARGUMENT	TYP	BESCHREIBUNG
`dir`	`cstring`	Verzeichnis für die Suche.

Beschreibung

Gibt den Dateinamen der ersten Datei oder des ersten Verzeichnisses im Verzeichnis `dir` auf der Speicherkarte zurück. Gibt eine leere Zeichenfolge zurück, wenn keine Datei oder keine Speicherkarte vorhanden ist. Diese Funktion kann mit der Funktion `FindFileNext` verwendet werden, um alle Dateien in allen Verzeichnissen zu durchsuchen.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
cstring
```

Beispiel

```
Name = FindFileFirst("/LOGS/LOG1/")
```

FindFileNext()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Gibt den Dateinamen der nächsten Datei oder des nächsten Verzeichnisses in dem Verzeichnis zurück, das in einem vorherigen Aufruf der Funktion `FindFileFirst` angegeben wurde. Gibt eine leere Zeichenfolge zurück, wenn keine weiteren Dateien vorhanden sind. Diese Funktion kann mit der Funktion `FindFileNext` verwendet werden, um alle Dateien in allen Verzeichnissen zu durchsuchen.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
cstring
```

Beispiel

```
Name = FindFileNext()
```

FindTagIndex(*label*)

ARGUMENT	TYP	BESCHREIBUNG
`label`	`cstring`	Die Tag-Kennzeichnung (nicht der Tag-Name oder die Gedächtnishilfe)

Beschreibung

Gibt die Indexnummer des mit `label` angegebenen Tags zurück.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

Beispiel

```
Index = FindTagIndex("Power")
```

Gibt die Indexnummer für das Tag mit der Kennzeichnung *Power* zurück.

Flash(*freq*)

ARGUMENT	TYP	BESCHREIBUNG
`freq`	`int`	Die Blinkhäufigkeit pro Sekunde.

Beschreibung

Gibt einen alternierenden Wert von “true” oder “false” zurück, der einen Zyklus `freq` Mal pro Sekunde durchläuft. Diese Funktion ist nützlich, wenn Sie Anzeigeprimitive animieren oder ihre Farben ändern.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Force(*dest, data*)

ARGUMENT	TYP	BESCHREIBUNG
`dest`	`int / float`	Das zu ändernde Tag.
`data`	`int / float`	Der zu schreibende Wert.

Beschreibung

Mit dieser Funktion wird das angegebene Tag auf den angegebenen Wert gesetzt. Sie unterscheidet sich dadurch von dem normalerweise verwendeten Zuweisungsoperator, dass sie (a) alle in der Warteschlange befindlichen Schreibvorgänge für dieses Tag löscht und sie durch einen sofortigen Schreibvorgang des angegebenen Werts ersetzt; und (b) einen Schreibvorgang auf das externe Kommunikationsgerät erzwingt, unabhängig davon, ob der Datenwert geändert wurde. Sie wird in Situationen verwendet, in denen das normale Schreibverhalten von Crimson nicht erforderlich ist.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

ForceCopy(*dest, src, count*)

ARGUMENT	TYP	BESCHREIBUNG
`dest`	`int / float`	Das erste Array-Element, in das kopiert werden soll.
`src`	`int / float`	Das erste Array-Element, aus dem kopiert werden soll.
`count`	`int`	Die Anzahl der zu verarbeitenden Elemente.

Beschreibung

Kopiert *count* (Anzahl) Array-Elemente aus `src` in `dest`. Die verwendete Semantik entspricht der Funktion `Force()`, wodurch die Schreibwarteschlange umgangen und ein Schreibvorgang erzwungen wird, unabhängig davon, ob die Originaldaten geändert wurden.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

ForceSQLSync()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Erzwingt die sofortige Ausführung des SQL-Synchronisierungsdienstes und überträgt Protokolldaten an den konfigurierten SQL-Server. Funktioniert nur, wenn die manuelle Synchronisierungseigenschaft des SQL-Synchronisierungsdienstes auf “Yes” gesetzt wurde.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
ForceSQLSync()
```

FormatCompactFlash()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Formatiert die Speicherkarte im Crimson-Gerät, wodurch alle Daten auf der Karte gelöscht werden. Sie sollten sicherstellen, dass der Benutzer eine entsprechende Warnung erhält, bevor diese Funktion aufgerufen wird.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
FormatCompactFlash()
```

FormatDrive(*drive*)

ARGUMENT	TYP	BESCHREIBUNG
`drive`	`int`	Der Buchstabe des Laufwerks, das formatiert werden soll.

Beschreibung

Formatiert ein an das System angeschlossenes Wechsellaufwerk und löscht alle darin enthaltenen Daten. Laufwerk C bezieht sich auf die Speicherkarte, während sich Laufwerk D auf den USB-Speicherstick bezieht.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
FormatDrive('C')
```

FtpGetFile(*server, loc, rem, delete*)

ARGUMENT	TYP	BESCHREIBUNG
`server`	`int`	Die FTP-Verbindungsnummer, immer 0.
`loc`	`cstring`	Der lokale Dateiname auf der Speicherkarte.
`rem`	`cstring`	Der Name der Datei auf dem FTP-Server.
`delete`	`int`	Wenn “true”, wird die Quelle nach der Übertragung gelöscht. Wenn “false”, verbleibt sie auf dem Quelldatenträger.

Beschreibung

Diese Funktion überträgt die definierte Datei vom FTP-Server auf die Speicherkarte des Crimson-Geräts. Gibt “true” zurück, wenn die Übertragung erfolgreich war, andernfalls “false”. Die Namen der Quell- und der Zieldatei können unterschiedlich sein. Der externe Pfad steht im Verhältnis zum Stammpfad des FTP-Servers. Weitere Informationen finden Sie im Synchronization Manager.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
Success = FtpGetFile(0, "/Recipes.csv", "/Recipes/Rec001.csv", 0)
```

FtpPutFile(*server, loc, rem, delete*)

ARGUMENT	TYP	BESCHREIBUNG
`server`	`int`	Die FTP-Verbindungsnummer, immer 0.
`loc`	`cstring`	Der lokale Dateiname auf der Speicherkarte.
`rem`	`cstring`	Der Name der Datei auf dem FTP-Server.
`delete`	`int`	Wenn “true”, wird die Quelle nach der Übertragung gelöscht. Wenn “false”, verbleibt sie auf dem Quelldatenträger.

Beschreibung

Diese Funktion überträgt die definierte Datei von der Speicherkarte des Crimson-Geräts auf den FTP-Server. Gibt “true” zurück, wenn die Übertragung erfolgreich war, andernfalls “false”. Die Namen der Quell- und der Zieldatei können unterschiedlich sein. Der externe Pfad steht im Verhältnis zum Root-Verzeichnis des FTP-Servers. Weitere Informationen finden Sie im Synchronization Manager.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiel

```
Success = FtpPutFile(0, "/LOGS/Report.txt", "/Reports/Report.txt", 1)
```

GetAlarmTag(*index*)

ARGUMENT	TYP	BESCHREIBUNG
`index`	`int`	Die Indexnummer des Tags.

Beschreibung

Diese Funktion gibt eine ganzzahlige Bitmaske zurück, die den Alarmstatus des Tags für das mit `index` gekennzeichnete Tag darstellt. Bit 0 (d. h. das Bit mit dem Wert 0x01) entspricht dem Status des Alarms 1 und Bit 1 (d. h. das Bit mit dem Wert 0x02) entspricht dem Status des Alarms 2. Der Index des Tags kann mit der Funktion `FindTagIndex()` im Namen des Tags gefunden werden, oder indem in der Konfigurationssoftware danach gesucht wird.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
AlarmsInTag = GetAlarmTag(12)
```

GetAutoCopyStatusCode()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Gibt einen Wert zurück, der den Status des Synchronisierungsvorgangs angibt, der optional auftreten kann, wenn ein USB-Stick in das Crimson-Gerät eingesetzt wird. Die möglichen Werte und deren Bedeutung:

WERT	BESCHREIBUNG
0	Synchronisierung ist nicht aktiviert.
1	Die Synchronisierungsaufgabe wird initialisiert.
2	Die Aufgabe wartet darauf, dass ein Speicherstick eingesetzt wird.
3	Die Aufgabe kopiert die erforderlichen Dateien.
4	Die Aufgabe ist abgeschlossen und wartet darauf, dass der Stick entfernt wird.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

GetAutoCopyStatusText()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Gibt eine Zeichenfolge zurück, die dem von `GetAutoCopyStatus()` zurückgegebenen Statuscode entspricht.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`cstring`

GetBatch()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Gibt den Namen des aktuellen Batches zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
cstring
```

Beispiel

```
CurrentBatch = GetBatch()
```

GetCameraData(*port, camera, param*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Die Nummer des Ports, an den die Kamera angeschlossen ist.
`camera`	`int`	Die Nummer der Kamera am Port.
`param`	`int`	Der zu lesende Kameraparameter.

Beschreibung

Diese Funktion gibt den Wert der Parameternummer von `param` für eine Banner Kamera zurück, die auf dem Crimson-Gerät verbunden ist. Das Argument `camera` ist die Gerätenummer, die in der Statusleiste von Crimson 3.1 angezeigt wird, wenn die Kamera ausgewählt ist. Unter dem Treiber kann mehr als eine Kamera angeschlossen werden. Die Zahl, die im Argument `port` platziert werden soll, ist die Portnummer, an die der Treiber gebunden ist. Weitere Informationen zu Parameternummern und -details finden Sie in der Banner-Dokumentation.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

Beispiel

```
Value = GetCameraData(4, 0, 1)
```

GetCurrentUserName()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Gibt den aktuellen Benutzernamen zurück, oder eine leere Zeichenfolge, falls kein Benutzer angemeldet ist. Beachten Sie, dass die Anzeige des aktuellen Benutzernamens die Sicherheit in Situationen beeinträchtigt, in denen Benutzernamen nicht allgemein bekannt sind. Daher sollte dies in Hochsicherheitsanwendungen mit Vorsicht eingesetzt werden.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`cstring`

GetCurrentUserRealName()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Gibt den echten Namen des aktuellen Benutzers oder eine leere Zeichenfolge zurück, falls kein Benutzer angemeldet ist.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`cstring`

GetCurrentUserRights()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Gibt die Benutzerrechte des aktuellen Benutzers zurück, wie für die Funktion `HasAccess()` definiert.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

GetDate (*time*) and Family

ARGUMENT	TYP	BESCHREIBUNG
`time`	`int`	Der zu decodierende Zeitwert.

Beschreibung

Jedes Mitglied dieser Funktionsfamilie gibt eine Komponente eines Zeit-/Datumswerts zurück, wie zuvor von `GetNow`, `Time` oder `Date` erstellt. Die verfügbaren Funktionen lauten wie folgt:

FUNCTION	BESCHREIBUNG
`GetDate`	Gibt den Tag des Monats in `time` zurück.
`GetDay`	Gibt den Wochentag in `time` zurück.
`GetDays`	Gibt die Anzahl der Tage in `time` zurück.
`GetHour`	Gibt die Stunden in `time` zurück.
`GetMin`	Gibt die Minuten in `time` zurück.
`GetMonth`	Gibt den Monat in `time` zurück.
`GetSec`	Gibt die Sekunden in `time` zurück.
`GetWeek`	Gibt die Kalenderwoche in `time` zurück.
`GetWeeks`	Gibt die Anzahl der Wochen in `time` zurück.
`GetWeekYear`	Gibt die Kalenderwoche zurück, wenn Wochennummern verwendet werden.
`GetYear`	Gibt das Jahr in `time` zurück.

Beachten Sie, dass `GetDays` und `GetWeeks` in der Regel zusammen mit dem Unterschied zwischen zwei Zeitwerten verwendet wird, um zu ermitteln, wie viel Zeit in Tagen oder Wochen verstrichen ist. Beachten Sie außerdem, dass das von `GetWeekYear` zurückgegebene Jahr nicht immer mit dem von `GetYear` zurückgegebenen Wert identisch ist, da der erste Wert einen kleineren Wert zurückgeben kann, wenn die letzte Woche eines Jahres über das Jahresende hinausgeht.

Funktionstyp

Diese Funktionen sind *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
d = GetDate(GetNow() - 12*60*60)
```

GetDeviceStatus(*device*)

ARGUMENT	TYP	BESCHREIBUNG
`device`	`int`	Das abzufragende Kommunikationsgerät.

Beschreibung

Gibt den Kommunikationsstatus des spezifischen Kommunikationsgeräts zurück.
Die beiden unteren Bits codieren den Fehlerstatus des Geräts wie folgt:

WERT	BESCHREIBUNG
0	Die Gerätekommunikation wird initialisiert.
1	Die Gerätekommunikation funktioniert ordnungsgemäß.
2	Die Gerätekommunikation weist einen oder mehrere “Soft Errors” auf.
3	Bei der Gerätekommunikation ist ein schwerwiegender Fehler aufgetreten.

Die folgenden Hexadezimalwerte codieren weitere Informationen zum Gerät:

WERT	BESCHREIBUNG
0x0010	In den automatischen Kommunikationsblöcken ist mindestens ein Fehler vorhanden.
0x0020	In den Gateway-Kommunikationsblöcken ist mindestens ein Fehler vorhanden.
0x0040	Die Kommunikation mit diesem Gerät ist unterbrochen.
0x0100	Eine (eingeschränkte) Rückmeldung wurde vom Gerät empfangen.
0x0200	Während der Kommunikation ist ein Fehler aufgetreten.
0x1000	Die primäre Schreibwarteschlange ist fast voll.
0x2000	Die sekundäre Schreibwarteschlange ist fast voll.

Beachten Sie, dass der Wert 0x0100 nicht impliziert, dass die Kommunikation richtig funktioniert, sondern lediglich, dass eine Rückmeldung empfangen wurde. Es ist hilfreich, die Verkabelung zu überprüfen usw. In ähnlicher Weise bedeutet der Wert 0x0200 nicht, dass die Kommunikation fehlgeschlagen ist, sondern zeigt an, dass nicht alles ganz reibungslos läuft. Der Wiederholungsmechanismus von Crimson kann beispielsweise eine Wiederherstellung nach Fehlern ermöglichen, sodass die Kommunikation scheinbar funktioniert. Das Bit kann jedoch immer noch darauf hinweisen, dass die Vorgänge nicht fehlerfrei laufen.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

GetDiskFreeBytes(*drive*)

ARGUMENT	TYP	BESCHREIBUNG
`drive`	`int`	Die Laufwerksnummer, immer 0.

Beschreibung

Gibt die Größe des freien Speicherplatzes in Kilobyte auf der Speicherkarte zurück. Beachten Sie, dass die Berechnung des verfügbaren Speicherplatzes erheblichen Aufwand erfordert, da viele Lesevorgänge durchgeführt werden müssen. Verwenden Sie diese Funktion daher nicht in einem Ausdruck, der zu häufig aufgerufen wird, indem Sie ihn auf einer Anzeigeseite platzieren oder als Reaktion auf ein Wahlereignis ausführen. Rufen Sie sie stattdessen als Reaktion auf das Ereignis `OnSelect` einer Seite auf und speichern Sie den Wert für die spätere Anzeige in einem Tag.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
FreeMemory = GetDiskFreeBytes(0)
```

GetDiskFreePercent(*drive*)

ARGUMENT	TYP	BESCHREIBUNG
`drive`	`int`	Die Laufwerksnummer, immer 0.

Beschreibung

Gibt den Prozentsatz des freien Speicherplatzes auf der Speicherkarte zurück. Beachten Sie, dass die Berechnung des verfügbaren Speicherplatzes erheblichen Aufwand erfordert, da viele Lesevorgänge durchgeführt werden müssen. Verwenden Sie diese Funktion daher nicht in einem Ausdruck, der zu häufig aufgerufen wird, indem Sie ihn auf einer Anzeigeseite platzieren oder als Reaktion auf ein Wahlereignis ausführen. Rufen Sie sie stattdessen als Reaktion auf das Ereignis `OnSelect` einer Seite auf und speichern Sie den Wert für die spätere Anzeige in einem Tag.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiel

```
FreeMemory = GetDiskFreePercent(0)
```

GetDiskSizeBytes(*drive*)

ARGUMENT	TYP	Beschreibung
`drive`	`int`	Die Laufwerksnummer, immer 0.

Beschreibung

Gibt die Größe der Speicherkarte in Kilobyte zurück. Beachten Sie, dass die Berechnung des verfügbaren Speicherplatzes erheblichen Aufwand erfordert, da viele Lesevorgänge durchgeführt werden müssen. Verwenden Sie diese Funktion daher nicht in einem Ausdruck, der zu häufig aufgerufen wird, indem Sie ihn auf einer Anzeigeseite platzieren oder als Reaktion auf ein Wahlereignis ausführen. Rufen Sie sie stattdessen als Reaktion auf das Ereignis `OnSelect` einer Seite auf und speichern Sie den Wert für die spätere Anzeige in einem Tag.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
CFSyze = GetDiskSizeBytes(0)
```

GetDriveStatus(*drive*)

ARGUMENT	TYP	BESCHREIBUNG
`drive`	`int`	Der Laufwerksbuchstabe des Laufwerks, das abgefragt werden soll.

Beschreibung

Gibt den Status des angegebenen Laufwerks wie folgt als Ganzzahl zurück:

WERT	STATUS	BESCHREIBUNG
0	Leer	Entweder ist keine Karte installiert oder die Karte wurde über einen Aufruf an die Funktion `DriveEject` ausgeworfen.
1	Ungültig	Die Karte ist beschädigt, falsch formatiert oder überhaupt nicht formatiert. Denken Sie daran, dass nur FAT16 unterstützt wird.
2	Wird geprüft	Die HMI prüft den Status der Karte. Dieser Status tritt ein, wenn eine Karte erstmals in die HMI eingesetzt wird.
3	Wird formatiert	Die HMI formatiert die Karte. Dieser Status tritt ein, wenn vom programmierenden PC ein Formatierungsvorgang angefordert wird.
4	Gesperrt	Entweder schreibt das Crimson-Gerät auf die Karte, oder die Karte ist angemeldet und Windows greift darauf zu.
5	Angemeldet	Eine gültige Karte ist installiert, aber sie ist weder durch das Crimson-Gerät noch durch Windows gesperrt.

Laufwerk C bezieht sich auf die Speicherkarte, während sich Laufwerk D auf den USB-Speicherstick bezieht.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

GetFileByte(*file*)

ARGUMENT	TYP	BESCHREIBUNG
`file`	`int`	Datei-Handle, das von `OpenFile` zurückgegeben wird.

Beschreibung

Liest ein einzelnes Byte aus der angegebenen Datei. Ein Wert von -1 zeigt das Ende der Datei an.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

GetFileData(*file, data, length*)

ARGUMENT	TYP	BESCHREIBUNG
`file`	`int`	Das Datei-Handle, das von `OpenFile` zurückgegeben wird.
`data`	`int`	Das erste Array-Element, in dem die Daten gespeichert werden sollen.
`length`	`int`	Die Anzahl der zu verarbeitenden Elemente.

Beschreibung

Liest die Bytes für *length* aus der angegebenen Datei und speichert sie in den angegebenen Array-Elementen. Der Rückgabewert gibt die Anzahl der erfolgreich gelesenen Bytes an und kann kleiner sein als *length*.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

GetFormattedTag(*index*)

ARGUMENT	TYP	BESCHREIBUNG
`index`	`int`	Indexnummer des Tags.

Beschreibung

Gibt eine Zeichenfolge zurück, die den formatierten Wert des durch `index` angegebenen Tags repräsentiert. Die zurückgegebene Zeichenfolge folgt dem auf dem Zieltag programmierten Format. Der Index kann mit der Funktion `FindTagIndex()` in der Tag-Bezeichnung gefunden werden, oder indem in der Crimson-Konfigurationssoftware danach gesucht wird. Diese Funktion ist mit allen Typen von Tags kompatibel.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
cstring
```

Beispiel

```
Value = GetFormattedTag(10)
```

GetInterfaceStatus(*port*)

ARGUMENT	TYP	BESCHREIBUNG
`interface`	`int`	The interface to be queried.

Beschreibung

Gibt eine Zeichenfolge zurück, die den Status der angegebenen TCP/IP-Schnittstelle angibt.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`cstring`

Beispiel

```
EthernetStatus = GetInterfaceStatus(1)
```

GetIntTag(*index*)

ARGUMENT	TYP	BESCHREIBUNG
`index`	`int`	Die Indexnummer des Tags.

Beschreibung

Gibt den Wert des ganzzahligen Tags zurück, angegebenen durch `index`. Der Index ist im Crimson-Konfigurationstool oder mithilfe der Funktion `FindTagIndex()` in der Tag-Bezeichnung zu finden. Diese Funktion funktioniert nur, wenn das Tag eine ganze Zahl ist.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
int
```

Beispiel

```
Value = GetIntTag(10)
```

GetLanguage()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Gibt die aktuell ausgewählte Sprache zurück, die an die Funktion `SetLanguage()` übergeben wurde.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

GetLastEventText(*all*)

ARGUMENT	TYP	BESCHREIBUNG
`all`	`int`	Auf “true” setzen, um auch Alarmereignisse einzuschließen.

Beschreibung

Gibt die Kennzeichnung des letzten Ereignisses zurück, das vom Ereignisprotokoll erfasst wurde. Wenn der Parameter `all` auf “true” gesetzt ist, enthält die Definition des Ereignisses auch diejenigen, die automatisch vom Alarmsystem generiert werden, und nicht nur die vom Ereignislogger generierten.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`cstring`

GetLastEventTime(*all*)

ARGUMENT	TYP	BESCHREIBUNG
`all`	`int`	Auf “true” setzen, um auch Alarmereignisse einzuschließen.

Beschreibung

Gibt den Zeitpunkt zurück, an dem die letzte Ereigniserfassung durch den Ereignislogger erfolgte. Der Wert kann unter Verwendung eines Felds mit Zeit- und Datumsformattyp in menschenlesbarer Form angezeigt werden. Wenn der Parameter `all` auf “true” gesetzt ist, enthält die Definition des Ereignisses auch Ereignisse, die automatisch vom Alarmsystem generiert werden, und nicht nur die vom Ereignislogger generierten.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

GetLastEventType(*all*)

ARGUMENT	TYP	BESCHREIBUNG
`all`	`int`	Auf “true” setzen, um auch Alarmereignisse einzuschließen.

Beschreibung

Gibt eine Zeichenfolge zurück, die den Typ des letzten Ereignisses angibt, das vom Ereignisprotokollierungssystem erfasst wurde. Wenn der Parameter `all` auf “true” gesetzt ist, enthält die Definition des Ereignisses auch Ereignisse, die automatisch vom Alarmsystem generiert werden, und nicht nur die vom Ereignislogger generierten.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`Cstring`

GetLastSQLSyncStatus()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Gibt den Status des letzten Zeitpunkts zurück, an dem der SQL-Synchronisierungsdienst versucht hat, die Datenprotokolle mit einem SQL-Server zu synchronisieren.

WERT	STATUS	BESCHREIBUNG
0	Ausstehend	Der Status des SQL-Synchronisierungsdienstes befindet sich in einem unbestimmten Status. Mögliche Gründe: Der Dienst muss noch ausgeführt werden, der Dienst hat die Synchronisierung mit dem SQL-Server noch nicht abgeschlossen oder der Dienst ist deaktiviert.
1	Erfolg	Der Dienst wurde erfolgreich mit dem SQL-Server synchronisiert.
2	Fehler	Der Dienst konnte nicht mit dem SQL-Server synchronisiert werden.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
Status = GetLastSQLSyncStatus()
```

GetLastSQLSyncTime(Request)

ARGUMENT	TYP	BESCHREIBUNG
`Request`	`int`	Die spezifische Zeit, die abgerufen werden soll.

Beschreibung

Gibt zurück, wann der SQL-Synchronisierungsdienst das letzte Mal mit einem SQL-Server synchronisiert wurde, seit das System hochgefahren wurde. Der zurückgegebene Wert ist für die Formatierung mit den Crimson-Zeitbearbeitungsfunktionen geeignet. Bis der Dienst versucht, eine Synchronisierung durchzuführen, werden alle drei Anforderungstypen als 0 zurückgegeben, was dem 1. Januar 1997 entspricht.

WERT	ANFORDERUNGS-TYP	BESCHREIBUNG
0	Letzte Startzeit	Ruft den letzten Zeitpunkt ab, zu dem der SQL-Synchronisierungsdienst eine Synchronisierung mit einem SQL-Server begann.
1	Letzte Erfolgszeit	Ruft den letzten Zeitpunkt ab, zu dem der Dienst erfolgreich eine Synchronisierung mit einem SQL-Server abgeschlossen hat.
2	Letzte Fehlerzeit	Ruft den letzten Zeitpunkt ab, zu dem der Dienst keine Synchronisierung mit einem SQL-Server durchführen konnte.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiele

```
LastStartTime = GetLastSQLSyncTime(0)
LastSuccessTime = GetLastSQLSyncTime(1)
LastFailTime = GetLastSQLSyncTime(2)
```

GetModelName(*code*)

ARGUMENT	TYP	BESCHREIBUNG
`code`	`int`	Der Name, der zurückgegeben werden soll. Derzeit wird nur 1 unterstützt.

Beschreibung

Gibt den Namen der Hardwareplattform zurück, auf der Crimson ausgeführt wird.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`cstring`

GetMonthDays(*y, m*)

ARGUMENT	TYP	BESCHREIBUNG
y	int	Das zu verarbeitende Jahr in vierstelliger Form.
m	int	Der zu verarbeitende Monat von 1 bis 12.

Beschreibung

Gibt die Anzahl der Tage im angegebenen Monat zurück, unter Berücksichtigung von Schaltjahren usw.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
Days = GetMonthDays(2000, 3)
```

GetNetGate(*port*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Der Index des Ethernet-Ports. Muss Null sein.

Beschreibung

Gibt die IP-Adresse des Standardgateways des Ports als Textzeichenfolge in Dezimalschreibweise zurück.

Funktionstyp

Die Funktion ist *passiv*.

Rückgabetyp

```
cstring
```

Beispiel

```
gate = GetNetGate(0)
```

GetNetId(*port*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Der Index des Ethernet-Ports.

Beschreibung

Meldet die MAC-Adresse eines Ethernet-Ports als 17-stellige Textzeichenfolge.

INDEX	BESCHREIBUNG
0	Gibt die Adresse des ersten oder einzigen konfigurierten Ports zurück.
1	Gibt Adresse von Port 1 zurück.
2	Gibt Adresse von Port 2 zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
cstring
```

Beispiel

```
MAC = GetNetId(1)
```

GetNetIp(*port*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Der Index des Ethernet-Ports.

Beschreibung

Meldet die IP-Adresse eines Ethernet-Ports als Textzeichenfolge in Dezimalschreibweise.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`cstring`

Beispiel

```
IP = GetNetIp(1)
```

GetNetMask(*port*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Der Index des Ethernet-Ports. Muss Null sein.

Beschreibung

Meldet die IP-Adressmaske eines Ethernet-Ports als Textzeichenfolge in Dezimalschreibweise.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`cstring`

Beispiel

```
mask = GetNetMask(0)
```

GetNow()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Gibt die aktuelle Uhrzeit und das aktuelle Datum als Anzahl der Sekunden zurück, die seit dem Bezugspunkt vom 1. Januar 1997 verstrichen sind. Dieser Wert kann dann mit anderen Zeit-/Datumsfunktionen verwendet werden.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
t = GetNow()
```

GetNowDate()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Gibt die Anzahl der Sekunden in den Tagen zurück, die seit dem 1. Januar 1997 vergangen sind.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
d = GetNowDate()
```

GetNowTime()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Gibt die Tageszeit in Sekunden zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
t = GetNowTime()
```

GetPortConfig(*port*, *param*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Die Nummer des festzulegenden Ports.
`param`	`int`	Der einzustellende Portparameter.

Beschreibung

Gibt den Wert eines Parameters am Port zurück. Die Portnummer beginnt bei der Programmierschnittstelle mit dem Wert 1. Die folgende Tabelle zeigt die verschiedenen Einstellungen für `param` und zugehörige Rückgabewerte.

WERT	PARAMETER	BESCHREIBUNG DES RÜCKGABEWERTS
`1`	Baudrate	Die tatsächliche Baudrate, z. B. 115200.
`2`	Datenbits	7, 8 oder 9.
`3`	Stoppbits	1 oder 2.
`4`	Parität	0keine 1ungerade 2gerade.
`5`	Physischer Modus	0RS-232, 1RS-422 Master, 2RS-422 Slave, 3RS-485.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

Beispiel

```
Port2Parity = GetPortConfig(2, 4)
```

GetRealTag(*index*)

ARGUMENT	TYP	BESCHREIBUNG
`index`	`int`	Indexnummer des Tags.

Beschreibung

Gibt den Wert des von `index` angegebenen tatsächlichen Tags zurück. Der Index kann mithilfe der Funktion `FindTagIndex()` aus der Tag-Bezeichnung abgerufen werden. Diese Funktion funktioniert nur, wenn das Tag eine reelle Zahl ist.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
float
```

Beispiel

```
Value = GetRealTag(10)
```

GetQueryStatus(*Query*)

ARGUMENT	TYP	BESCHREIBUNG
`query`	`string`	Der Name der Abfrage, wie in der SQL Manager-Baumstruktur angegeben.

Beschreibung

Ruft ab, wann die angegebene Abfrage zuletzt versucht wurde.

WERT	STATUS	BESCHREIBUNG
`0`	`Pending`	Der Status der Abfrage befindet ist unbestimmt. Entweder ist die Abfrage noch auszuführen oder sie wird aktiv ausgeführt.
`1`	`Erfolg`	Die Abfrage wurde erfolgreich ausgeführt und es wurden Daten für alle Spalten abgerufen.
`2`	`Partial Data`	Die Abfrage hat einige Daten abgerufen, jedoch nicht für alle Spalten. Überprüfen Sie, ob der konfigurierte Spaltentyp mit dem tatsächlichen Typ der Spalte in der SQL-Datenbank übereinstimmt.
`3`	`Failure`	Die Abfrage konnte keine Daten abrufen.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
Status := GetQueryStatus("Query1")
```

GetQueryTime(*Query*)

ARGUMENT	TYP	BESCHREIBUNG
`query`	`string`	Der Name der Abfrage, wie in der SQL Manager-Baumstruktur angegeben.

Beschreibung

Ruft ab, wann die Ausführung der angegebenen Abfrage zuletzt versucht wurde. Wenn die Abfrage nicht wie erwartet ausgeführt wurde, vergewissern Sie sich, dass der SQL Manager verbunden ist und dass das Netzwerk ordnungsgemäß konfiguriert ist.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
Time := GetQueryTime("Query1")
```

GetRestartCode(*n*)

ARGUMENT	TYP	BESCHREIBUNG
`n`	`int`	Der Eintrag in der Neustarttabelle, von 0 bis einschließlich 6.

Beschreibung

Gibt den Guru Meditation Code entsprechend dem angegebenen Neustart zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`cstring`

GetRestartInfo(*n*)

ARGUMENT	TYP	BESCHREIBUNG
`n`	`int`	Der Eintrag in der Neustarttabelle, von 0 bis einschließlich 6.

Beschreibung

Gibt eine erweiterte Beschreibung des angegebenen Neustarts mit Zeit- und Datumstempel zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`cstring`

GetRestartText(*n*)

ARGUMENT	TYP	BESCHREIBUNG
`n`	`int`	Der Eintrag in der Neustarttabelle, von 0 bis einschließlich 6.

Beschreibung

Gibt eine erweiterte Beschreibung des angegebenen Neustarts ohne Zeit- und Datumstempel zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`cstring`

GetRestartTime(*n*)

ARGUMENT	TYP	BESCHREIBUNG
`n`	`int`	Der Eintrag in der Neustarttabelle, von 0 bis einschließlich 6.

Beschreibung

Gibt die Uhrzeit zurück, zu der der angegebene Neustart stattgefunden hat. Nicht für alle Situationen definiert.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

GetSQLConnectionStatus()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Ruft den Status der SQL Manager-Verbindung ab.

WERT	STATUS	BESCHREIBUNG
0	Pending	Der Status der SQL Manager-Verbindung ist unbestimmt. Mögliche Gründe: Die Verwaltungsvorgänge müssen noch ausgeführt werden, der Manager fragt aktiv Daten ab oder der Manager ist deaktiviert.
1	Erfolg	Der Manager hat erfolgreich eine Verbindung mit dem SQL-Server hergestellt.
2	Failure	Der Manager konnte keine Verbindung mit dem externen SQL-Server herstellen. Vergewissern Sie sich, dass das Netzwerk richtig konfiguriert ist und dass die Anmeldedaten korrekt sind.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
Status := GetSQLConnectionStatus()
```

GetStringTag(*index*)

ARGUMENT	TYP	BESCHREIBUNG
`index`	`int`	Indexnummer des Tags.

Beschreibung

Gibt den Wert des von `index` angegebenen Zeichenfolgen-Tags zurück. Der Index kann mithilfe der Funktion `FindTagIndex()` aus der Tag-Bezeichnung abgerufen werden. Diese Funktion funktioniert nur, wenn das Tag eine Zeichenfolge ist.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
cstring
```

Beispiel

```
Value = GetStringTag(10)
```

GetTagLabel(*index*)

ARGUMENT	TYP	BESCHREIBUNG
`index`	`int`	Indexnummer des Tags.

Beschreibung

Gibt die Kennzeichnung des von `index` angegebenen Tags zurück.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
cstring
```

Beispiel

```
Label = GetTagLabel(10)
```

GetUpDownData(*data, limit*)

ARGUMENT	TYP	BESCHREIBUNG
`data`	`int`	Ein stetig steigender Quellwert.
`limit`	`int`	Die Anzahl der zu generierenden Werte.

Beschreibung

Diese Funktion nimmt einen stetig zunehmenden Wert und konvertiert diesen in einen Wert, der zwischen 0 und `limit`-1 pendelt. Sie wird in der Regel in einer Demodatenbank verwendet, um eine realistisch wirkende Animation zu erzeugen. Häufig wird dabei `DispCount` als Parameter für `data` übergeben, sodass sich der resultierende Wert bei jeder Bildschirmaktualisierung ändert. Wenn die Funktion `GetUpDownStep` mit denselben Argumenten aufgerufen wird, wird ein Wert zurückgegeben, der die Änderungsrichtung der von `GetUpDownData` zurückgegebenen Daten anzeigt.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiel

```
Data = GetUpDownData(DispCount, 100)
```

GetUpDownStep(*data, limit*)

ARGUMENT	TYP	BESCHREIBUNG
`data`	`int`	Ein stetig steigender Quellwert.
`limit`	`int`	Die Anzahl der zu generierenden Werte.

Beschreibung

Eine Beschreibung dieser Funktion finden Sie unter `GetUpDownData`.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiel

```
Delta = GetUpDownStep(DispCount, 100)
```

GetVersionInfo(*code*)

ARGUMENT	TYP	BESCHREIBUNG
`code`	`int`	Das zurückzugebende Element.

Beschreibung

Gibt wie folgt Informationen zu den verschiedenen Versionsnummern zurück:

CODE	BESCHREIBUNG
1	Gibt die Bootloader-Version zurück.
2	Gibt den Build der Laufzeitsoftware zurück.
3	Gibt den Build der Konfigurationssoftware zurück, die für die Vorbereitung der aktuellen Datenbank verwendet wurde.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

GetWebParamHex(*param*)

ARGUMENT	TYP	BESCHREIBUNG
`param`	`cstring`	Der abzurufende Parameter.

Beschreibung

Ruft einen Parameter ab, der über die URL-Abfragezeichenfolge an die benutzerdefinierte Webseite übergeben wird. Der Parameter wird als Hexadezimalzahl interpretiert und dann zurückgegeben. Diese Funktion wird in der Regel in Code verwendet und über die eingebettete Tag-Syntax der benutzerdefinierten Website aufgerufen.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
Value1 = GetWebParamHex("Count");
```

GetWebParamInt(*param*)

ARGUMENT	TYP	BESCHREIBUNG
`param`	`cstring`	Der abzurufende Parameter.

Beschreibung

Ruft einen Parameter ab, der über die URL-Abfragezeichenfolge an die benutzerdefinierte Webseite übergeben wird. Der Parameter wird als Dezimalzahl interpretiert und dann zurückgegeben. Diese Funktion wird in der Regel in Code verwendet und über die eingebettete Tag-Syntax der benutzerdefinierten Website aufgerufen.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
Value1 = GetWebParamInt("Count")
```

GetWebParamString(*param*)

ARGUMENT	TYP	BESCHREIBUNG
`param`	`cstring`	Der abzurufende Parameter.

Beschreibung

Ruft einen Parameter ab, der über die URL-Abfragezeichenfolge an die benutzerdefinierte Webseite übergeben wird. Der Parameter wird als Zeichenfolge zurückgegeben, die die im Parameter übergebenen Zeichen enthält.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
cstring
```

Beispiel

```
Value1 = GetWebParamHex("Test")
```

GotoNext()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Veranlasst, dass das Fenster wieder in den Seitenverlaufspuffer verschoben wird, wodurch das Ergebnis eines vorherigen Aufrufs an `GotoPrevious()` umgekehrt wird. Der Teil des Verlaufspuffers, der über diese Funktion zugänglich ist, wird gelöscht, wenn die Funktion `GotoPage()` aufgerufen wird.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
GotoNext()
```

GotoPage(*name*)

ARGUMENT	TYP	BESCHREIBUNG
`name`	Bildschirmseite	Die anzuzeigende Seite.

Beschreibung

Wählt die Seite `name` zur Anzeige auf dem Display des Crimson-Geräts aus.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
GotoPage(Page1)
```

GotoPrevious()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Weist das Bedienfeld an, zur letzten Seite zurückzukehren, die auf dem Display des Crimson-Geräts angezeigt wird. Die Seite wird aus einem Verlaufspuffer extrahiert, sodass sich “Vorherige” auf die zuvor angezeigte Seite bezieht, nicht auf die vorherige Seite im Navigationsfenster der Anzeigeseite.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
GotoPrevious()
```

GreaterEqR64(*a*, *b*)

ARGUMENT	TYP	BESCHREIBUNG
`a`	`int`	Der zu vergleichende Wert.
`b`	`int`	Der Wert, mit dem verglichen wird.

Beschreibung

Vergleicht den Wert von `a` mit `b` mithilfe von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und gibt 1 zurück, wenn *a* größer oder gleich *b* ist, andernfalls 0. Dies entspricht a >= b mit doppelter Genauigkeit. Die Eingabeoperanden *a* und *b* sollten von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu `AddR64` ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
Result = GreaterEqR64(a[0], b[0])
```

GreaterR64(*a*, *b*)

ARGUMENT	TYP	BESCHREIBUNG
a	int	Der zu vergleichende Wert.
b	int	Der Wert, mit dem verglichen wird.

Beschreibung

Vergleicht den Wert von `a` mit `b` mithilfe von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und gibt 1 zurück, wenn *a* größer als *b* ist, andernfalls 0. Dies entspricht a > b mit doppelter Genauigkeit. Die Eingabeoperanden *a* und *b* sollten von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu `AddR64` ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiel

```
Result = GreaterR64(a[0], b[0])
```

HasAccess (*rights*)

ARGUMENT	TYP	BESCHREIBUNG
`rights`	`int`	Die erforderlichen Zugriffsrechte.

Beschreibung

Gibt den Wert **true** oder **false** zurück, je nachdem, ob der aktuelle Benutzer Zugriffsrechte besitzt, die durch den Parameter `rights` definiert sind. Dieser Parameter enthält eine Bitmaske, um die verschiedenen benutzerdefinierten Rechte darzustellen, wobei das Bit 0 (d. h. das Bit mit dem Wert 0x01) für das Benutzerrecht 1 steht, Bit 1 (d. h. das Bit mit dem Wert 0x02) für das Benutzerrecht 2 und so weiter. Die Funktion wird in der Regel in Programmen verwendet, die Aktionen ausführen, die möglicherweise sicherheitsrelevant sind.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiel

```
if( HasAccess(1)) {
Data1 = 0;
Data2 = 0;
Data3 = 0;
}
```

HasAllAccess(*rights*)

ARGUMENT	TYP	BESCHREIBUNG
`rights`	`int`	Die erforderlichen Zugriffsrechte.

Beschreibung

Gibt den Wert **true** oder **false** zurück, je nachdem, ob der aktuelle Benutzer alle Zugriffsrechte besitzt, die durch den Parameter `rights` definiert sind. Dieser Parameter enthält eine Bitmaske, um die verschiedenen benutzerdefinierten Rechte darzustellen, wobei das Bit 0 (d. h. das Bit mit dem Wert 0x01) für das Benutzerrecht 1 steht, Bit 1 (d. h. das Bit mit dem Wert 0x02) für das Benutzerrecht 2 und so weiter. Die Funktion wird in der Regel in Programmen verwendet, die Aktionen ausführen, die möglicherweise sicherheitsrelevant sind.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

HideAllPopups()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Blendet alle Popups aus, einschließlich verschachtelter Popups, angezeigt durch `ShowPopup()` oder `ShowNested()`.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

HidePopup()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Blendet das Popup aus, das zuvor mit `ShowPopup` angezeigt wurde.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
HidePopup()
```

IncR64(*result, tag*)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag`	`int`	Der zu verarbeitende Wert.

Beschreibung

Erhöht den Wert von `tag` um Eins unter Verwendung von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in `result`. Dies entspricht dem Operator ++ mit doppelter Genauigkeit. Der Eingabeoperand *tag* sollte von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu AddR64 ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`void`

Beispiel

```
IncR64(result[0], tag[0])
```

IntToR64(*result, n*)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`n`	`int`	Der Wert, der konvertiert werden soll.

Beschreibung

Konvertiert den Wert, der in `n` gespeichert ist, aus einer Ganzzahl in eine 64-Bit-Gleitkommazahl mit doppelter Genauigkeit und speichert das Ergebnis in `result`. Das Tag *result* muss ein Eintrag in einem ganzzahligen Array sein, sodass von diesem Punkt aus auf mindestens zwei Register zugegriffen werden kann. Nach der Ausführung dieser Funktion ist der in *result* gespeicherte Wert für die Verwendung in anderen mathematischen 64-Bit-Funktionen geeignet. Ein Beispiel für die beabsichtigte Verwendung dieser Funktion finden Sie im Eintrag zu AddR64.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`void`

Beispiel

```
IntToR64(result[0], n)
```

IntToText(*data*, *radix*, *count*)

ARGUMENT	TYP	BESCHREIBUNG
`data`	`int`	Der zu verarbeitende Wert.
`radix`	`int`	Die zu verwendende Zahlenbasis.
`count`	`int`	Die Anzahl der zu generierenden Ziffern.

Beschreibung

Gibt die Zeichenfolge zurück, die durch die Formatierung von `data` in der Basiswurzel `radix` ermittelt wurde, wobei `count` Ziffern generiert werden. Der Wert wird ohne Vorzeichen betrachtet. Wenn also ein Wert mit Vorzeichen erforderlich ist, verwenden Sie Sgn, um zu entscheiden, ob ein negatives Vorzeichen vorangestellt werden und dann `Abs` verwendet werden soll, um den absoluten Wert an `IntToText` zu übergeben.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`cstring`

Beispiel

```
PortPrint(1, IntToText(Value, 10, 4))
```

IsBatchNameValid(*name*)

ARGUMENT	TYP	BESCHREIBUNG
`name`	`cstring`	Der zu testende Batch-Name.

Beschreibung

Gibt *true* zurück, wenn der angegebene Batch-Name gültige Zeichen enthält und nicht bereits vorhanden ist.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

IsBatteryLow()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Gibt *true* zurück, wenn die interne Batterie des Geräts schwach ist.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

IsDeviceOnline(*device*)

ARGUMENT	TYP	BESCHREIBUNG
`device`	`int`	Der Index des zu prüfenden Geräts.

Beschreibung

Meldet, ob das Gerät `device` online ist oder nicht. Ein Gerät wird als offline markiert, wenn eine wiederholte Folge von Kommunikationsfehlern aufgetreten ist. Wenn sich ein Gerät im Offlinestatus befindet, wird es regelmäßig abgefragt, um zu sehen, ob es wieder online ist.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
Okay = IsDeviceOnline(1)
```

IsLoggingActive()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Gibt “true” oder “false” zurück, um anzugeben, ob die Datenprotokollierung in der aktuellen Datenbank aktiv ist. Der Wert “true” gibt an, dass ein Protokoll definiert wurde und dass das Protokoll mindestens ein Daten-Tag enthält.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

IsPortRemote(*port*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Der abzufragende Kommunikationsport.

Beschreibung

Gibt *true* zurück, wenn der angegebene Port über die Portfreigabe übernommen wurde.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

IsSQLSyncRunning()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Gibt an, ob der SQL-Synchronisierungsdienst aktuell versucht, mit einem SQL-Server zu synchronisieren.

WERT	STATUS	BESCHREIBUNG
0	Wird nicht ausgeführt	Der SQL-Synchronisierungsdienst führt keine Synchronisierung mit einem SQL-Server durch.
1	Wird ausgeführt	Der Dienst führt derzeit eine Synchronisierung mit einem SQL-Server durch.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
IsRunning = IsSQLSyncRunning()
```

IsWriteQueueEmpty(*dev*)

ARGUMENT	TYP	BESCHREIBUNG
`dev`	`int`	Die Gerätenummer, für die der Warteschlangenstatus abgerufen werden soll.

Beschreibung

Gibt den Status der Schreibwarteschlange für das mit dem Argument *dev* gekennzeichnete Gerät zurück. Die Funktion gibt “true” zurück, wenn die Warteschlange leer ist, andernfalls “false”. Die Gerätenummer kann in der Statusleiste von Crimson identifiziert werden, wenn in “Kommunikation” ein Gerät ausgewählt ist. Wenn ein Kommunikationsfehler auftritt, während die Schreibwarteschlange nicht leer ist oder wenn während eines solchen Fehlers Daten geschrieben werden, wird die Warteschlange geleert, aber die zu schreibenden Daten stehen weiterhin aus. Die ausstehenden Daten werden geschrieben, sobald der Kommunikationsfehler behoben wurde. Daher kann mit dieser Funktion nicht zuverlässig festgestellt werden, ob für ein Gerät Schreibvorgänge ausstehen, wenn Kommunikationsfehler auf dem Gerät vorhanden sind.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiel

```
QueueEmpty = IsWriteQueueEmpty(1)
```

KillDirectory(*name*)

ARGUMENT	TYP	BESCHREIBUNG
`name`	`cstring`	Das Verzeichnis, das gelöscht werden soll.

Beschreibung

Löscht das angegebene Verzeichnis und alle darin enthaltenen Unterverzeichnisse oder Dateien; gibt *true* zurück, wenn die Funktion erfolgreich ist, oder *false*, wenn die Funktion fehlschlägt.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

Left(*string, count*)

ARGUMENT	TYP	BESCHREIBUNG
`string`	`cstring`	Die zu bearbeitende Zeichenfolge.
`count`	`int`	Die Anzahl der zurückzugebenden Zeichen.

Beschreibung

Gibt die ersten `count` (Anzahl) Zeichen aus `string` zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`cstring`

Beispiel

```
AreaCode = Left(Phone, 3)
```

Len(*string*)

ARGUMENT	TYP	BESCHREIBUNG
`string`	`cstring`	Die zu bearbeitende Zeichenfolge.

Beschreibung

Gibt die Anzahl der Zeichen in `string` zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiel

```
Size = Len(Input)
```

LessEqR64(*a*, *b*)

ARGUMENT	TYP	BESCHREIBUNG
`a`	`int`	Der zu vergleichende Wert.
`b`	`int`	Der Wert, mit dem verglichen wird.

Beschreibung

Vergleicht den Wert von `a` mit `b` mithilfe von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und gibt 1 zurück, wenn *a* kleiner als oder gleich *b* ist, andernfalls 0. Dies entspricht a <= b mit doppelter Genauigkeit. Die Eingabeoperanden *a* und *b* sollten von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu `AddR64` ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
Result = LessEqR64(a[0], b[0])
```

LessR64(a, b)

ARGUMENT	TYP	BESCHREIBUNG
a	int	Der zu vergleichende Wert.
b	int	Der Wert, mit dem verglichen wird.

Beschreibung

Vergleicht den Wert von `a` mit `b` mithilfe von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und gibt 1 zurück, wenn *a* kleiner als *b* ist, andernfalls 0. Dies entspricht a < b mit doppelter Genauigkeit. Die Eingabeoperanden *a* und *b* sollten von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu AddR64 ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
LessR64(a[0], b[0])
```

LoadCameraSetup(*port, camera, index, file*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Die Nummer des Ports, an den die Kamera angeschlossen ist.
`camera`	`int`	Die Gerätenummer der Kamera.
`index`	`int`	Die Prüfdateinummer in der Kamera.
`file`	`cstring`	Pfad und Dateiname für die Prüfdatei.

Beschreibung

Diese Funktion lädt die Prüfdatei von der Speicherkarte des Crimson-Geräts in den Kameraspeicher. Die Zahl, die im Argument `port` platziert werden soll, ist die Portnummer, an die der Treiber gebunden ist. Das Argument `camera` ist die Gerätenummer, die in der Statusleiste von Crimson 3.1 angezeigt wird, wenn die Kamera ausgewählt ist. Unter einem Treiber kann mehr als eine Kamera angeschlossen werden. Der `index` stellt die Prüfdateinummer in der Kamera dar, in der die Datei geladen wird. `file` steht für Pfad und Dateiname der Quellprüfdatei auf der Speicherkarte. Die Funktion gibt “true” zurück, wenn die Übertragung erfolgreich ist, andernfalls “false”. Beachten Sie, dass diese Funktion am besten in einem Benutzerprogramm aufgerufen wird, das im Hintergrund läuft, sodass die HMI genügend Zeit für den Zugriff auf die Speicherkarte hat.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

Beispiel

```
Success = LoadCameraSetup(4, 0, 1, “in0.isp”)
```

LoadSecurityDatabase(*mode, file*)

ARGUMENT	TYP	BESCHREIBUNG
`mode`	`int`	Das zu verwendende Dateiformat.
`file`	`cstring`	Die Datei, die die Datenbank enthalten soll.

Beschreibung

Lädt die Sicherheitsdatenbank der Datenbank aus der angegebenen Datei. Mit dem Wert 1 für *mode* werden die Benutzerliste mit Benutzernamen, echten Namen und Passwörtern gespeichert und geladen. In jedem Fall wird die Datei verschlüsselt und enthält keine Klartextpasswörter.

Der Rückgabewert ist *true* bei Erfolg oder *false* bei einem Fehler.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

Log(*value*)

ARGUMENT	TYP	BESCHREIBUNG
`value`	`float`	Der zu verarbeitende Wert.

Beschreibung

Gibt den natürlichen Logarithmus für `value` zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`float`

Beispiel

```
Variable1 = log(5.0)
```

Log_{10}(*value*)

ARGUMENT	TYP	BESCHREIBUNG
`value`	`float`	Der zu verarbeitende Wert.

Beschreibung

Gibt den Logarithmus zur Basis 10 für `value` zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`float`

Beispiel

```
Variable3 = log10(5.0)
```

Log10R64(result, tag)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag`	`int`	Der zu verarbeitende Wert.

Beschreibung

Berechnet den Logarithmus zur Basis 10 von *tag* unter Verwendung von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in *result*. Der Eingabeoperand *tag* sollte von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu `AddR64` ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`void`

Beispiel

```
Log10R64(result[0], tag1[0])
```

LogBatchComment(*set*, *text*)

ARGUMENT	TYP	BESCHREIBUNG
`set`	`int`	Die Batch-Setnummer.
`text`	`cstring`	Der Kommentar, der protokolliert werden soll.

Beschreibung

Protokolliert einen Kommentar zu allen Batches, die mit dem angegebenen Batch-Set verknüpft sind.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

LogBatchHeader(*set, text*)

ARGUMENT	TYP	BESCHREIBUNG
`set`	`int`	Die Batch-Setnummer.
`text`	`cstring`	Der zu protokollierende Header.

Beschreibung

Protokolliert einen Header-Kommentar zu allen Batches, die mit dem angegebenen Batch-Set verknüpft sind. Der Aufruf sollte unmittelbar nach dem Erstellen eines neuen Batches erfolgen. Die Kommentare werden immer vor allen anderen Daten in der Datei platziert.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

LogComment(log, text)

ARGUMENT	TYP	BESCHREIBUNG
`log`	`int`	Der Index des Protokolls, auf das zugegriffen werden soll.
`text`	`cstring`	Der Textkommentar, der dem Protokoll hinzugefügt werden soll.

Beschreibung

Fügt einen Kommentar zu einem Datenprotokoll hinzu. Das Datenprotokoll muss so konfiguriert sein, dass Kommentare über die entsprechende Eigenschaft unterstützt werden. Mithilfe von Kommentaren können Batchoder andere Details zu Beginn eines Protokolls bereitgestellt werden, oder der Bediener kann während der Protokollierung einen Punkt markieren, der Aufmerksamkeit erfordert.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
int
```

Beispiel

```
LogComment(1, "Start of Shift")
```

LogHeader(*log, text*)

ARGUMENT	TYP	BESCHREIBUNG
`log`	`int`	Der Index oder der Name des Protokolls.
`text`	`cstring`	Der Kommentar, der protokolliert werden soll.

Beschreibung

Zeichnet einen Kommentar in der angegebenen Protokolldatei auf. Für das fragliche Protokoll müssen Kommentare aktiviert sein.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

logR64(*result, tag*)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag`	`int`	Der zu verarbeitende Wert.

Beschreibung

Berechnet den natürlichen Logarithmus von tag unter Verwendung von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in *result*. Der Eingabeoperand tag sollte von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu AddR64 ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
void
```

Beispiel

```
logR64(result[0], tag[0])
```

LogSave()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Zwingt den Datenlogger, auf der Speicherkarte zu speichern. Diese Funktion sollte **nicht** regelmäßig aufgerufen werden. Sie ist nur für die punktuelle Verwendung vorgesehen. Eine übermäßige Nutzung dieser Funktion kann dazu führen, dass die Speicherkarte beschädigt wird und Daten verloren gehen.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
LogSave()
```

MakeFloat(*value*)

ARGUMENT	TYP	BESCHREIBUNG
`value`	`int`	Der Wert, der konvertiert werden soll.

Beschreibung

Interpretiert das Ganzzahlargument als Gleitkommawert neu. Diese Funktion führt **keine** Typenkonvertierung durch, sondern nimmt das im Argument gespeicherte Bitmuster und geht davon aus, dass es statt einer Ganzzahl eine Gleitkommazahl darstellt. Sie kann dazu verwendet werden, Daten von einem externen Gerät zu bearbeiten, das möglicherweise einen anderen Datentyp aufweist, als vom Kommunikationstreiber erwartet wird.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`float`

Beispiel

```
fp = MakeFloat(n)
```

MakeInt(*value*)

ARGUMENT	TYP	BESCHREIBUNG
`value`	`float`	Der Wert, der konvertiert werden soll.

Beschreibung

Interpretiert das Gleitkommaargument als Ganzzahl neu. Diese Funktion führt **keine** Typenkonvertierung durch, sondern nimmt das im Argument gespeicherte Bitmuster und geht davon aus, dass es statt eines Gleitkommawerts eine Ganzzahl darstellt. Sie kann dazu verwendet werden, Daten von einem externen Gerät zu bearbeiten, das möglicherweise einen anderen Datentyp aufweist, als vom Kommunikationstreiber erwartet wird.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
n = MakeInt(fp)
```

Max(*a*, *b*)

ARGUMENT	TYP	BESCHREIBUNG
`a`	`int / float`	Der erste Wert, der verglichen werden soll.
`b`	`int / float`	Der zweite Wert, der verglichen werden soll.

Beschreibung

Gibt das größere der beiden Argumente zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int` oder `float`, je nach Typ der Argumente.

Beispiel

```
Larger = Max(Tank1, Tank2)
```

MaxR64(result, tag1, tag2)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag1`	`int`	Der erste Wert, der verglichen werden soll.
`tag2`	`int`	Der zweite Wert, der verglichen werden soll.

Beschreibung

Berechnet den größeren Wert von *tag1* und *tag2* mithilfe von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in *result*. Die Eingabeoperanden *tag1* und *tag2* sollten von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu `AddR64` ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`void`

Beispiel

```
MaxR64(result[0], tag1[0], tag2[0])
```

MaxU32(*tag1*, *tag2*)

ARGUMENT	TYP	BESCHREIBUNG
`tag1`	`int`	Der erste Wert, der verglichen werden soll.
`tag2`	`int`	Der zweite Wert, der verglichen werden soll.

Beschreibung

Gibt den größeren Wert von *tag1* und *tag2* in einem Kontext ohne Vorzeichen zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiel

```
Larger = MaxU32(tag1, tag2)
```

Mean(*element, count*)

ARGUMENT	TYP	BESCHREIBUNG
`element`	`int/float`	Das erste Array-Element, das verarbeitet werden soll.
`count`	`int`	Die Anzahl der zu verarbeitenden Elemente.

Beschreibung

Gibt den Mittelwert der `count` (Anzahl) Array-Elemente ab `element` zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`float`

Beispiel

```
Average = Mean(Data[0], 10)
```

Mid(*string, pos, count*)

ARGUMENT	TYP	BESCHREIBUNG
`string`	`cstring`	Die zu bearbeitende Zeichenfolge.
`pos`	`int`	Die Position, an der begonnen werden soll.
`count`	`int`	Die Anzahl der zurückzugebenden Zeichen.

Beschreibung

Gibt `count` (Anzahl) Zeichen ab Position `pos` innerhalb von `string` zurück, wobei 0 die erste Position ist.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`cstring`

Beispiel

```
Exchange = Mid(Phone, 3, 3)
```

Min(*a*, *b*)

ARGUMENT	TYP	BESCHREIBUNG
`a`	`int / float`	Der erste Wert, der verglichen werden soll.
`b`	`int / float`	Der zweite Wert, der verglichen werden soll.

Beschreibung

Gibt das kleinere der beiden Argumente zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int` oder `float`, je nach Typ der Argumente.

Beispiel

```
Smaller = Min(Tank1, Tank2)
```

MinR64(*result, tag1, tag2*)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag1`	`int`	Der erste zu vergleichende Wert.
`tag2`	`int`	Der zweite zu vergleichende Wert.

Beschreibung

Berechnet den kleineren Wert von *tag1* und *tag2* mithilfe von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in *result*. Die Eingabeoperanden *tag1* und *tag2* sollten von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu `AddR64` ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
void
```

Beispiel

```
MinR64(result[0], tag1[0], tag2[0])
```

MinU32(*tag1, tag2*)

ARGUMENT	TYP	BESCHREIBUNG
`tag1`	`int`	Der erste Wert, der verglichen werden soll.
`tag2`	`int`	Der zweite Wert, der verglichen werden soll.

Beschreibung

Gibt den kleineren Wert von *tag1* und *tag2* in einem Kontext ohne Vorzeichen zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiel

```
Smaller = MinU32(tag1, tag2)
```

MinusR64(*result, tag*)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag`	`int`	Der zu verarbeitende Wert.

Beschreibung

Kehrt das Vorzeichen von *tag* unter Verwendung von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit um und speichert das Ergebnis in *result*. Diese Funktion bewirkt, dass positive Zahlen negativ und negative Zahlen positiv werden. Der Eingabeoperand *tag* sollte von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu `AddR64` ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`void`

Beispiel

```
MinusR64(result[0], tag[0])
```

ModU32(*tag1*, *tag2*)

ARGUMENT	TYP	Beschreibung
`tag1`	`int`	Der Dividend.
`tag2`	`int`	Der Divisor.

Beschreibung

Gibt den Wert von *tag1* modulo *tag2* in einem Kontext ohne Vorzeichen zurück. Dies ist das Äquivalent von *tag1 % tag2* ohne Vorzeichen oder der Rest von tag1 dividiert durch tag2.

Funktionstyp

Diese Funktion ist **MinU32(*tag1*, *tag2*)**.

ARGUMENT	TYPE	BESCHREIBUNG
`tag1`	`int`	Der erste Wert, der verglichen werden soll.
`tag2`	`int`	Der zweite Wert, der verglichen werden soll.

Rückgabetyp

`int`

Beispiel

```
Result = ModU32(tag1, tag2)
```

MountCompactFlash(*enable*)

ARGUMENT	TYP	BESCHREIBUNG
`enable`	`int`	Der gewünschte Mount (Lade)- oder Dismount (Entlade)-Zustand.

Beschreibung

Lädt oder entlädt die Speicherkarte als Laufwerk, auf das über Windows Explorer zugegriffen werden kann. Wenn *enable* auf 1 gesetzt ist, wird die Karte geladen. Wenn *enable* auf 0 gesetzt ist, wird die Karte entladen. Das Gerät wird nach dem Aufruf dieser Funktion automatisch neu gestartet. Diese Funktion stellt dieselbe Funktionalität wie die Optionen “Mount Flash” und “Dismount Flash” im Menü “Link” der Crimson 3.1-Software bereit.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
void
```

Beispiel

```
MountCompactFlash(0)
```

MoveFiles(*source, target, flags*)

ARGUMENT	TYP	BESCHREIBUNG
`source`	`cstring`	Der Pfad, aus dem die Dateien verschoben werden sollen.
`target`	`cstring`	Der Pfad, in den die Dateien verschoben werden sollen.
`flags`	`int`	Die Markierungen, die den Verschiebevorgang steuern.

Beschreibung

Verschiebt alle Dateien aus dem Verzeichnis *source* in das Verzeichnis *target*.
Die verschiedenen Bits in `flags` ändern den Verschiebevorgang wie folgt:

BIT	GEWICHT	BESCHREIBUNG
0	1	Wenn diese Option festgelegt ist, wird der Vorgang rekursiv in Unterverzeichnissen durchgeführt.
1	2	Wenn festgelegt, werden vorhandene Dateien überschrieben. Wenn leer, bleiben die vorhandenen Dateien unverändert.

Der Rückgabewert der Funktion ist bei Erfolg *true* und bei einem Fehler *false*.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

MulDiv(*a*, *b*, *c*)

ARGUMENT	TYP	BESCHREIBUNG
a	int	Der erste Wert.
b	int	Der zweite Wert.
c	int	Der dritte Wert.

Beschreibung

Gibt `a*b/c` zurück. Die Zwischenberechnungen erfolgen mit 64-Bit-Ganzzahlen, um Überläufe zu vermeiden.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
d = MulDiv(a, b, c)
```

MulR64(*result, tag1, tag2*)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag1`	`int`	Der Multiplikand.
`tag2`	`int`	Der Multiplikator.

Beschreibung

Berechnet den Wert von *tag1* mal *tag2* mithilfe von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in *result*. Dies entspricht *tag1 * tag2* mit doppelter Genauigkeit. Die Eingabeoperanden *tag1* und *tag2* sollten von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Unter `AddR64` ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`void`

Beispiel

```
MulR64(result[0], tag1[0], tag2[0])
```

MulU32(*tag1, tag2*)

ARGUMENT	TYP	BESCHREIBUNG
`tag1`	`int`	Das Multiplikanden-Tag.
`tag2`	`int`	Das Multiplikator-Tag.

Beschreibung

Gibt den Wert von *tag1* mal *tag2* in einem Kontext ohne Vorzeichen zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
Result = MulU32(tag1, tag2)
```

MuteSiren()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Schaltet die innere Sirene des Crimson-Geräts aus.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
MuteSiren()
```

NetworkPing(*address, timeout*)

ARGUMENT	TYP	BESCHREIBUNG
`address`	`int`	Die Zieladresse zum Senden einer Ping-Anfrage.
`timeout`	`int`	Die Zeit in Millisekunden, die auf eine Antwort gewartet werden soll.

Beschreibung

Sendet eine ICMP-Echoanforderung (im Allgemeinen als Ping bezeichnet) an die angegebene IP-Adresse und wartet die angegebene Timeout-Zeit auf eine Antwort. Der Parameter “timeout” muss in Millisekunden angegeben werden. Wenn innerhalb der Timeout-Zeit eine gültige Antwort empfangen wird, gibt die Funktion 1 zurück. Wenn innerhalb der Timeout-Zeit keine Antwort empfangen wird oder diese Funktion aufgerufen wird, während der Ethernet-Port deaktiviert ist, gibt die Funktion 0 zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiele

```
IP = TextToAddr("192.168.1.100");
Result = NetworkPing(IP, 5000);
IP = ResolveDNS("redlion.net");
Result = NetworkPing(IP, 5000);
```

NewBatch(*name*)

ARGUMENT	TYP	BESCHREIBUNG
`name`	`cstring`	Der Batch-Name.

Beschreibung

Startet einen Batch mit dem Namen `name`. Der Name darf höchstens 8 Zeichen lang sein und muss aus Zeichen für einen gültigen FAT16-Dateinamen bestehen. Durch einen Neustart eines Batches, der sich bereits auf der CF-Karte befindet, werden die Daten angehängt. Wenn ein neuer Batch die maximale Anzahl zu bewahrender Batches überschreitet, wird der älteste Batch gelöscht. Wenn der Name leer ist, entspricht die Funktion `EndBatch()`. Der Status des Batches wird während eines Aus- und Einschaltvorgangs beibehalten. Beachten Sie, dass das Starten eines neuen Batches innerhalb von weniger als 10 Sekunden nach Beendigung oder Start des letzten Batches zu nicht definiertem Verhalten führt. Um direkt von einem Batch zu einem anderen zu wechseln, rufen Sie `NewBatch()` ohne einen dazwischen liegenden Aufruf von `EndBatch()` auf.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
NewBatch("ProdA")
```

Nop()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Diese Funktion hat keine Auswirkungen.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
Nop()
```

NotEqualR64(*a*, *b*)

ARGUMENT	TYP	BESCHREIBUNG
`a`	`int`	Der erste Wert, der verglichen werden soll.
`b`	`int`	Der zweite Wert, der verglichen werden soll.

Beschreibung

Vergleicht den Wert von *a* mit *b* mithilfe von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und gibt 1 zurück, wenn *a* ungleich *b* ist, andernfalls 0. Dies entspricht `a!=b` mit doppelter Genauigkeit. Der Vergleich von Gleitkommawerten für exakte Gleichheit ist aufgrund von Rundungsfehlern fehleranfällig. Die Eingabeoperanden *a* und *b* sollten von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu `AddR64` ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiel

```
NotEqualR64(a[0], b[0])
```

OpenFile(*name*, *mode*)

ARGUMENT	TYP	BESCHREIBUNG
`name`	`cstring`	Die zu öffnende Datei.
`mode`	`int`	Der Modus, in dem die Datei geöffnet werden soll: 0 = Schreibgeschützt 1 = Lesen/Schreiben am Anfang der Datei 2 = Lesen/Schreiben am Ende der Datei

Beschreibung

Gibt ein Handle an die Datei `name` auf der Speicherkarte zurück. Diese Funktion ist auf maximal vier geöffnete Dateien beschränkt. Die Speicherkarte kann nicht entladen werden, während eine Datei geöffnet ist. Das Ablagesystem von Crimson 3.1 unterstützt jetzt sowohl FAT16 als auch FAT32. Wenn die Speicherkarte mit FAT32 formatiert wurde, können lange Dateinamen verwendet werden. Wenn die Speicherkarte mit FAT16 formatiert wurde, werden lange Dateinamen nicht unterstützt. Beachten Sie, dass bei Verwendung von umgekehrten Schrägstrichen im Pfadnamen zur Trennung von Pfadelementen diese gemäß den Crimson-Regeln für Zeichenfolgenkonstanten verdoppelt werden müssen, wie im Crimson 3.1 User Manual im Kapitel zum Schreiben von Ausdrücken beschrieben. Um diese Komplikation zu vermeiden, können Schrägstriche anstelle von umgekehrten Schrägstrichen verwendet werden. Schrägstriche müssen nicht verdoppelt werden. Beachten Sie, dass diese Funktion keine Datei erstellt, die nicht existiert. Rufen Sie dazu `CreateFile()` auf, bevor Sie diese Funktion aufrufen.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
int
```

Beispiel

```
hFile = OpenFile("/LOGS/LOG1/01010101.csv", 0)
```

Pi()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Gibt *pi* als Gleitkommazahl zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`float`

Beispiel

```
Scale = Pi()/180
```

PlayRTTTL(*tune*)

ARGUMENT	TYP	BESCHREIBUNG
`tune`	`cstring`	Die Melodie, die in der RTTTL-Darstellung wiedergegeben werden soll.

Beschreibung

Gibt mit dem internen Signaltongeber des Crimson-Geräts eine Melodie wieder. Das Argument `tune` sollte die Melodie im RTTTL-Format enthalten: das Format, das von einer Reihe von Mobiltelefonen für benutzerdefinierte Klingeltöne verwendet wird. Beispielmusik kann von vielen Websites im World Wide Web heruntergeladen werden.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
PlayRTTTL("TooSexy:d=4,o=5,b=40:16f,16g,16f,16g,16f.,16f,16g,16f,16g,16g#.,16g#,16g
,16g#,16g,
16f.,16f,16g,16f,16g,16f.,16f,16g,16f,16g,16f.,16f,16g,16f,16g,16g#.,16g#,16g,16g#,16
g,16f.,1 6f,16g,16f,16g,32f.")
```

PopDev(*element, count*)

ARGUMENT	TYP	BESCHREIBUNG
`element`	`int / float`	Das erste Array-Element, das verarbeitet werden soll.
`count`	`int`	Die Anzahl der zu verarbeitenden Elemente.

Beschreibung

Gibt die Standardabweichung von `count` (Anzahl) Array-Elementen ab `element` zurück, davon ausgehend, dass die Datenpunkte die gesamte untersuchte Population darstellen. Wenn Sie die Standardabweichung in einer Stichprobe ermitteln müssen, verwenden Sie stattdessen die Funktion `StdDev`.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
float
```

Beispiel

```
Dev = PopDev(Data[0], 10)
```

PortClose(*port*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Schließt den angegebenen Port.

Beschreibung

Diese Funktion wird zusammen mit den aktiven oder passiven TCP RAW-Port-Treibern verwendet, um den ausgewählten Port zu schließen, indem die Verbindung mit dem zugehörigen Socket ordnungsgemäß geschlossen wird.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
PortClose(6)
```

PortGetCTS(*port*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Der RAW-Port, von dem der CTS-Status abgerufen werden soll.

Beschreibung

Gibt den Status der CTS-Leitung am seriellen Port zurück, der durch `port` angegeben ist. Der Port muss so konfiguriert sein, dass er einen RAW-Treiber verwendet. Die Nummer des Kommunikationsports kann in der Statusleiste von Crimson angezeigt werden, wenn der Port ausgewählt ist.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
int
```

Beispiel

```
CtsState = PortGetCTS(2)
```

PortInput(*port, start, end, timeout, length*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Der zu lesende RAW-Port.
`start`	`int`	Das abzugleichende Anfangszeichen, falls vorhanden.
`end`	`int`	Das abzugleichende Endzeichen, falls vorhanden.
`timeout`	`int`	Die Zeitüberschreitung zwischen Zeichen in Millisekunden, falls vorhanden.
`length`	`int`	Die maximale Anzahl der zu lesenden Zeichen, falls vorhanden.

Beschreibung

Liest eine Zeichenfolge aus dem von `port` angegebenen Port, wobei die verschiedenen anderen Parameter zur Steuerung des Eingabeprozesses verwendet werden. Wenn `start` nicht Null ist, wartet der Prozess, bis das durch diesen Parameter angegebene Zeichen empfangen wird. Wenn `start` Null ist, beginnt der Empfangsprozess sofort. Der Prozess wird dann fortgesetzt, bis eine der folgenden Bedingungen erfüllt ist:

- `end` ist nicht Null, und ein Zeichen, das mit `end` übereinstimmt, wird empfangen.
- `timeout` ist nicht Null, und dieser Zeitraum vergeht, ohne dass Zeichen empfangen werden.
- `length` ist nicht Null, und die entsprechende Anzahl an Zeichen wurde empfangen.

Die Funktion gibt dann die empfangenen Zeichen zurück, ohne des `start`- oder `end`-Bytes. Bei einer Zeitüberschreitung werden die empfangenen Zeichen nur dann zurückgegeben, wenn sowohl der Parameter “end” als auch der Parameter “length” Null ist. Wenn die Parameter “end” und/oder “length” nicht Null sind, gibt die Funktion eine leere Zeichenfolge zurück. Diese Funktion wird zusammen mit RAW-Port-Treibern verwendet, um benutzerdefinierte Protokolle mithilfe der Crimson-Programmiersprache zu implementieren. Sie ersetzt die RYOP-Funktionalität in Edict.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
cstring
```

Beispiel

```
Frame = PortInput(1, '*', 13, 100, 200)
```

PortPrint(*port, string*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Der RAW-Port, an den geschrieben werden soll.
`string`	`cstring`	Die zu übertragende Textzeichenfolge.

Beschreibung

Sendet den in `string` enthaltenen Text an den durch `port` angegebenen Port. Der Port muss so konfiguriert sein, dass er einen RAW-Treiber verwenden kann, wie z. B. den seriellen RAW-Port-Treiber oder einen der RAW-TCP/IP-Treiber. Die Daten werden übertragen, und die Funktion gibt zurück. Der Port-Treiber verarbeitet bei Bedarf den Handshake und die Steuerung der Senderaktivierungsleitungen.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
PortPrint(1, "ABCD")
```

PortPrintEx(*port, string*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Der RAW-Port, an den geschrieben werden soll.
`string`	`string`	String

Beschreibung

Sendet den in `string` enthaltenen Text an den durch `port` angegebenen Port. Der Port muss so konfiguriert sein, dass er einen RAW-Treiber verwenden kann, wie z. B. den seriellen RAW-Port-Treiber oder einen der RAW-TCP/IP-Treiber. Die Daten werden übertragen, und die Funktion gibt zurück. Der Port-Treiber verarbeitet bei Bedarf den Handshake und die Steuerung der Senderaktivierungsleitungen. Beim Senden von Daten über einen TCP/IP-RAW-Port wird versucht, die Daten in einem einzigen Paket zu senden, wenn möglich, oder so wenige Pakete wie möglich zu verwenden.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
PortPrintEx(4, "ABCDEFGHIJKLMNOPQRSTUVWXYZ");
```

PortRead(*port, period*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Der zu lesende RAW-Port.
`period`	`int`	Die Wartezeit in Millisekunden.

Beschreibung

Versucht, ein Zeichen aus dem durch `port` angegebenen Port zu lesen. Der Port muss so konfiguriert sein, dass er einen RAW-Treiber verwenden kann, wie z. B. den seriellen RAW-Port-Treiber oder einen der RAW-TCP/IP-Treiber. Wenn innerhalb des angegebenen Zeitraums keine Daten verfügbar sind, wird der Wert -1 zurückgegeben. Die Einstellung von `period` auf Null führt dazu, dass alle Daten in der Warteschlange zurückgegeben werden, verhindert jedoch, dass Crimson auf Daten wartet, wenn keine verfügbar ist.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

Beispiel

```
Data = PortRead(1, 100)
```

PortSendData(port, data, count)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Der RAW-Port, an den geschrieben werden soll.
`data`	`int`	Das erste Element des Arrays der zu sendenden Daten.
`count`	`int`	Die Anzahl der zu sendenden Elemente.

Beschreibung

Überträgt `count` (Anzahl) Elemente des Arrays, beginnend ab `data`, an den Port, der durch `port` angegeben ist. Der Port muss so konfiguriert sein, dass er einen RAW-Treiber verwenden kann, wie z. B. den seriellen RAW-Port-Treiber oder einen der RAW-TCP/IP-Treiber. Die Daten werden übertragen, und die Funktion gibt zurück. Der Port-Treiber verarbeitet bei Bedarf den Handshake und die Steuerung der Senderaktivierungsleitungen. Über einen TCP/IP-RAW-Port gesendete Daten werden in einem einzigen Paket gesendet, falls möglich. Jedes Element im Eingabe-Array *data* sollte ein Byte der gewünschten Daten darstellen, die gesendet werden sollen. Die Elemente im Array können einen beliebigen Wert zwischen 0 und 255 annehmen. Diese Funktion bietet eine Alternative zur textbasierten Funktion `PortPrint`, sodass binäre Daten gesendet werden können.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
PortSendData(4, Data[0], 16);
```

PortSetRTS(*port, state*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Der zu steuernde RAW-Port.
`state`	`int`	Der Status von RTS: “true” (1) oder “false” (0).

Beschreibung

Legt den Status der RTS-Leitung am seriellen Port fest, der durch *port* angegeben ist. Der Port muss so konfiguriert werden, dass er einen RAW-Treiber verwendet, und er muss einer der seriellen Ports sein. Das Statusargument kann nur die Werte 0 oder 1 annehmen. Die Nummer des Ports wird in der Statusleiste von Crimson 3.1 angezeigt, wenn der Port ausgewählt ist.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
PortSetRTS(2, 1)
```

PortWrite(*port, data*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Der RAW-Port, an den geschrieben werden soll.
`data`	`int`	Das zu übertragende Byte.

Beschreibung

Sendet das durch `data` angegebene Byte an den durch `port` angegebenen Port. Der Port muss so konfiguriert sein, dass er einen RAW-Treiber verwenden kann, wie z. B. den seriellen RAW-Port-Treiber oder einen der RAW-TCP/IP-Treiber. Das Zeichen wird übertragen, und die Funktion gibt zurück. Der Port-Treiber verarbeitet bei Bedarf den Handshake und die Steuerung der Senderaktivierungsleitungen.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
PortWrite(1, 'A')
```

PostKey(*code, transition*)

ARGUMENT	TYP	BESCHREIBUNG
`code`	`int`	Tastencode.
`transition`	`int`	Übergangscode.

Beschreibung

Fügt der Eingabewarteschlange eine physische Tastenbedienung hinzu.

CODE	TASTE
`0x80`	Soft Key 1
`0x81`	Soft Key 2
`0x82`	Soft Key 3
`0x83`	Soft Key 4
`0x84`	Soft Key 5
`0x85`	Soft Key 6
`0x86`	Soft Key 7
`0x90`	Funktionstaste 1
`0x91`	Funktionstaste 2
`0x92`	Funktionstaste 3
`0x93`	Funktionstaste 4
`0x94`	Funktionstaste 5

CODE	TASTE
`0x95`	Funktionstaste 6
`0x96`	Funktionstaste 7
`0x97`	Funktionstaste 8
`0xA0`	ALARME
`0xA1`	STUMM
`0x1B`	BEENDEN
`0xA2`	MENÜ
`0xA3`	ANHEBEN
`0xA4`	ABSENKEN
`0x09`	NÄCHST.
`0x08`	VORH.
`0x0D`	EINGABE

ÜBERGANG	VORGANG
`0`	Taste nach unten und dann Taste nach oben senden.
`1`	Nur Taste nach unten senden.
`2`	Nur Taste nach oben senden.
`3`	Nur Tastenwiederholung senden.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`void`

Beispiel

```
PostKey(0x80, 0)
```

Power(*value*, *power*)

ARGUMENT	TYP	BESCHREIBUNG
`value`	`int / float`	Der zu verarbeitende Wert.
`power`	`int / float`	Die Potenz, in der `value` berechnet werden soll.

Beschreibung

Gibt `value` hoch `power` zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int` oder `float`, abhängig vom Typ des Arguments für `value`.

Beispiel

```
Volume = Power(Length, 3)
```

PowR64(*result, value, power*)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`value`	`int`	Der zu verarbeitende Wert.
`power`	`int`	Die Potenz, in der *value* berechnet wird.

Beschreibung

Berechnet den Wert von *value* hoch *power* unter Verwendung von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in *result*. Die Eingabeoperanden *value* und *power* sollten von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu `AddR64` ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`void`

Beispiel

```
PowR64(result[0], tag1[0], tag2[0])
```

PrintScreenToFile(*path*, *name*, *res*)

ARGUMENT	TYP	BESCHREIBUNG
`path`	`cstring`	Das Verzeichnis, in dem die Datei erstellt werden soll.
`name`	`cstring`	Der zu verwendende Dateiname.
`res`	`int`	Die erforderliche Farbauflösung des Bildes.

Beschreibung

Speichert eine Bitmap-Kopie der aktuellen Anzeige in der angegebenen Datei. Wenn Sie eine leere Zeichenfolge für *name* übergeben, kann Crimson einen eindeutigen Dateinamen für das neue Bild auswählen. Das *res*-Argument kann auf Eins gesetzt werden, um eine Bitmap mit 8 Bit pro Pixel zu erstellen, während bei einem Wert von Null eine Bitmap mit 16 Bit pro Pixel erstellt wird. Der letztere Wert erzeugt wesentlich größere Dateien, da diese Dateien keine RLE8-Komprimierung unterstützen können. Der Rückgabewert gibt an, ob die Funktion erfolgreich war.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

PutFileByte(*file, data*)

ARGUMENT	TYP	BESCHREIBUNG
`file`	`int`	Das Datei-Handle, das von `OpenFile` zurückgegeben wird.
`data`	`int`	Der zu schreibende Datenwert.

Beschreibung

Schreibt ein einzelnes Byte in die angegebene Datei. Gibt 1 für Erfolg und -1 für Fehler zurück.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

PutFileData(*file, data, length*)

ARGUMENT	TYP	BESCHREIBUNG
`file`	`int`	Das Datei-Handle, das von `OpenFile` zurückgegeben wird.
`data`	`int`	Das erste Array-Element, das geschrieben werden soll.
`length`	`int`	Die Anzahl der zu verarbeitenden Elemente.

Beschreibung

Schreibt eine bestimmte Anzahl von Bytes in die Datei, wobei von jedem Array-Element ein Byte übernommen wird.

Der Rückgabewert ist die Anzahl der geschriebenen Bytes und darf kleiner als *length* sein.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

R64ToInt(x)

ARGUMENT	TYP	BESCHREIBUNG
`x`	`int`	Der Wert, der konvertiert werden soll.

Beschreibung

Konvertiert den 64-Bit-Gleitkommawert mit doppelter Genauigkeit, der in x als Array mit dem Umfang 2 gespeichert ist, als Ganzzahl mit Vorzeichen und gibt das Ergebnis zurück. In der Regel enthält das Array x einen 64-Bit-Gleitkommawert, der als Ergebnis von einer der bereitgestellten mathematischen 64-Bit-Funktionen ermittelt wurde. Die durch das Array *x* dargestellte Zahl muss als 32-Bit-Ganzzahl darstellbar sein, damit diese Konvertierung erfolgreich durchgeführt wird. Weitere Informationen finden Sie im Eintrag zu `AddR64`.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiel

```
Result = R64ToInt(x[0])
```

R64ToReal(x)

ARGUMENT	TYP	BESCHREIBUNG
x	int	Der Wert, der konvertiert werden soll.

Beschreibung

Konvertiert den 64-Bit-Gleitkommawert mit doppelter Genauigkeit, der in *x* als Array mit dem Umfang 2 gespeichert ist, als 32-Bit-Gleitkommazahl und gibt das Ergebnis zurück. In der Regel enthält das Array *x* einen 64-Bit-Gleitkommawert, der als Ergebnis von einer der bereitgestellten mathematischen 64-Bit-Funktionen ermittelt wurde. Die durch das Array *x* dargestellte Zahl muss als 32-Bit-Gleitkommazahl darstellbar sein, damit diese Konvertierung erfolgreich durchgeführt wird. Ein Beispiel für die Verwendung der mathematischen 64-Bit-Funktionen finden Sie im Eintrag zu `AddR64`.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
float
```

Beispiel

```
Result = R64ToReal(x[0])
```

Rad2Deg(*theta*)

ARGUMENT	TYP	BESCHREIBUNG
`theta`	`float`	Der zu verarbeitende Winkel.

Beschreibung

Gibt `theta` zurück, konvertiert von Radianten in Grad.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`float`

Beispiel

```
Right = Rad2Deg(Pi()/2)
```

Random(*range*)

ARGUMENT	TYP	BESCHREIBUNG
`range`	`int`	Der Bereich der zu erzeugenden Zufallswerte.

Beschreibung

Gibt einen Pseudo-Zufallswert zwischen 0 und `range-1` zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
Noise = Random(100)
```

ReadData(*data, count*)

ARGUMENT	TYP	BESCHREIBUNG
`data`	`any`	Das erste Array-Element, das gelesen werden soll.
`count`	`int`	Die Anzahl der zu lesenden Elemente.

Beschreibung

Fordert, dass bei der nächsten Kommunikationsprüfung `count` (Anzahl) Elemente ab dem Array-Element `data` gelesen werden. Diese Funktion wird bei Arrays verwendet, die externen Daten zugeordnet wurden und deren Leserichtlinie auf *manuell lesen* gesetzt wurde. Die Funktion kehrt sofort zurück und wartet nicht auf die zu lesenden Daten.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
ReadData(array1[8], 10)
```

ReadFile(*file, chars*)

ARGUMENT	TYP	BESCHREIBUNG
`file`	`int`	Das Datei-Handle, wie es von OpenFile gefordert wird.
`chars`	`int`	Die Anzahl der zu lesenden Zeichen.

Beschreibung

Liest eine Zeichenfolge mit bis zu 512 Zeichen Länge aus der angegebenen Datei. Diese Funktion sucht nicht nach Zeilenvorschub oder Wagenrücklauf, sondern liest eine bestimmte Anzahl von Bytes. Die von `ReadFile()` zurückgegebene Zeichenfolge enthält die erforderliche Anzahl von Zeilen, um die Anzahl der zu lesenden Zeichen zu erreichen. Zeilenvorschub und Wagenrücklauf sind Teil der zurückgegebenen Zeichenfolge.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
string
```

Beispiel

```
Text = ReadFile(hFile, 80)
```

ReadFileLine(*file*)

ARGUMENT	TYP	BESCHREIBUNG
`file`	`int`	Das Datei-Handle, das von `OpenFile` zurückgegeben wird.

Beschreibung

Gibt eine einzelne Textzeile aus der Datei zurück.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`cstring`

Beispiel

```
Text = ReadFileLine(hFile)
```

RealToR64(*result, n*)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`n`	`float`	Der Wert, der konvertiert werden soll.

Beschreibung

Konvertiert den in n aus einer realen Zahl gespeicherten Wert in eine 64-Bit-Zahl mit doppelter Genauigkeit und speichert das Ergebnis als Array mit der Länge 2 in *result*. Das Tag *result* muss daher ein ganzzahliges Array mit einem Umfang von mindestens 2 sein. Nach der Ausführung dieser Funktion ist der in *result* gespeicherte Wert für die Verwendung in anderen mathematischen 64-Bit-Funktionen geeignet. Ein Beispiel für die beabsichtigte Verwendung dieser Funktion finden Sie im Eintrag zu `AddR64`.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
void
```

Beispiel

```
RealToR64(result[0], n)
```

RenameFile(*handle, name*)

ARGUMENT	TYP	BESCHREIBUNG
`handle`	`int`	Das Datei-Handle.
`name`	`cstring`	Der neue Dateiname.

Beschreibung

Gibt einen Wert von nicht Null zurück, wenn die Datei erfolgreich umbenannt wurde. Das Datei-Handle ist der zurückgegebene Wert der Funktion `Openfile()`. Nach dem Umbenennen bleibt die Datei geöffnet und sollte geschlossen werden, wenn keine weiteren Vorgänge erforderlich sind. Der Dateiname darf maximal 8 Zeichen lang sein, ohne die Erweiterung, die maximal 3 Zeichen lang sein darf.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

Beispiel

```
Result = RenameFile(File , "NewName.txt")
```

ResolveDNS(*name*)

ARGUMENT	TYP	BESCHREIBUNG
`name`	`cstring`	Der DNS-Name, der aufgelöst werden soll.

Beschreibung

Gibt die IP-Adresse des angegebenen DNS-Namens zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
ip = ResolveDNS("www.redlion.net")
```

Right(*string, count*)

ARGUMENT	TYP	BESCHREIBUNG
`string`	`cstring`	Die zu bearbeitende Zeichenfolge.
`count`	`int`	Die Anzahl der zurückzugebenden Zeichen.

Beschreibung

Gibt die letzten `count` (Anzahl) Zeichen aus `string` zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`cstring`

Beispiel

```
Local = Right(Phone, 7)
```

RShU32(tag1, tag2)

ARGUMENT	TYP	BESCHREIBUNG
`tag1`	`int`	Der Wert, der verschoben werden soll.
`tag2`	`int`	Der Betrag, um den verschoben werden soll.

Beschreibung

Gibt den Wert von *tag1* um *tag2* Bits nach rechts verschoben in einem Kontext ohne Vorzeichen zurück. Dies entspricht *tag1 >> tag2* ohne Vorzeichen.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiel

```
Shifted = RShU32(tag1, tag2)
```

RunAllQueries()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Weist den SQL Manager an, alle konfigurierten Abfragen auszuführen.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
RunAllQueries()
```

RunQuery(query)

ARGUMENT	TYP	BESCHREIBUNG
`query`	`string`	Der Name der Abfrage, wie in der SQL Manager-Baumstruktur angegeben.

Beschreibung

Weist den SQL Manager an, die angegebene Abfrage auszuführen.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
RunQuery("Query1")
```

RxCAN(*port, data, id*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Der RAW-CAN-Port
`data`	`int`	Das erste Array-Element, in dem die empfangenen Daten gespeichert werden sollen.
`id`	`int`	29-Bit CAN-Kennung.

Beschreibung

Ruft empfangene CAN-Nachrichten ab, die mit RxCANInit initialisiert wurden. Die ersten vier Bytes der empfangenen Nachricht werden im angegebenen Array-Element gepackt (Big Endian), während die restlichen Bytes (sofern vorhanden) im folgenden Element des Arrays gespeichert werden. Gibt bei Erfolg einen Wert von 1 oder bei einem Fehler 0 zurück.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

Beispiel

```
RxCAN(8, Data, 0x12345678)
```

RxCANInit(*port, id, dlc*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Der RAW-CAN-Port.
`id`	`int`	29-Bit CAN-Kennung.
`dlc`	`int`	Datenlänge 1 bis 8 Byte.

Beschreibung

Initialisiert die programmgesteuerte Übertragung von CAN-Nachrichten über eine CAN-Optionskarte. Die Funktion gibt bei Erfolg einen Wert von 1 oder bei einem Fehler 0 zurück. Aufrufe sollten erst nach dem Start des Systems erfolgen, und jede 29-Bit-Kennung darf nur einmal initialisiert werden.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
int
```

Beispiel

```
RxCANInit(8, 0x12345678, 8)
```

SaveCameraSetup(*port, camera, index, file*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Die Nummer des Ports, an den die Kamera angeschlossen ist.
`camera`	`int`	Die Gerätenummer der Kamera.
`index`	`int`	Die Prüfdateinummer in der Kamera.
`file`	`cstring`	Pfad und Dateiname für die Prüfdatei.

Beschreibung

Beachten Sie, dass diese Funktion die Prüfdatei, die von der Kamera hochgeladen wurde, auf der Speicherkarte Crimson-Geräts speichert. Die Zahl, die im Argument `port` platziert werden soll, ist die Portnummer, an die der Treiber gebunden ist. Das Argument `camera` ist die Gerätenummer, die in der Statusleiste von Crimson 3.1 angezeigt wird, wenn die Kamera ausgewählt ist. Unter einem Treiber kann mehr als eine Kamera angeschlossen werden. Der `index` steht für die Prüfdateinummer in der Kamera. `file` entspricht dem Pfad und Dateinamen, in dem die Prüfdatei auf der Speicherkarte gespeichert werden soll. Diese Funktion gibt “true” zurück, wenn die Übertragung erfolgreich ist, andernfalls “false”. Diese Funktion sollte in einem Benutzerprogramm aufgerufen werden, das im Hintergrund ausgeführt wird, damit die HMI genügend Zeit für den Zugriff auf die Speicherkarte hat.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
int
```

Beispiel

```
Success = SaveCameraSetup(4, 0, 1, "\\in0.isp")
```

SaveConfigFile(*file*)

ARGUMENT	TYP	BESCHREIBUNG
`file`	`cstring`	Die Abbilddatei, in die geschrieben werden soll.

Beschreibung

Speichern Sie den aktuellen Bootloader, die Firmware und das Datenbank-Abbild für die anschließende Übertragung an ein anderes Gerät in einer CI3-Datei. Beachten Sie, dass Abbilddateien, die auf diese Weise erstellt wurden, nur die Firmware für genau das Hardwaremodell enthalten, auf dem sie erstellt wurden, und möglicherweise nicht mit ähnlichen, aber nicht identischen Geräten funktionieren. Der Rückgabewert ist *true* bei Erfolg oder *false* bei einem Fehler.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
int
```

Beispiel

```
SaveConfigFile("image.ci3")
```

SaveSecurityDatabase(*mode*, *file*)

ARGUMENT	TYP	BESCHREIBUNG
`mode`	`int`	Das zu verwendende Dateiformat.
`file`	`cstring`	Die Datei, die die Datenbank enthalten soll.

Beschreibung

Speichert die Sicherheitsdatenbank der Datenbank aus der angegebenen Datei. Ein *mode*-Wert von 0 wird zum Speichern verwendet und lädt anschließend nur das mit jedem Benutzer verknüpfte Passwort. Mit dem Wert 1 für *mode* werden die Benutzerliste mit Benutzernamen, echten Namen und Passwörtern gespeichert und geladen. In jedem Fall wird die Datei verschlüsselt und enthält keine Klartextpasswörter. Der Rückgabewert ist *true* bei Erfolg oder *false* bei einem Fehler.

Rückgabetyp

`int`

Scale(*data*, r_1, r_2, e_1, e_2)

ARGUMENT	TYP	BESCHREIBUNG
`data`	`int`	Der Wert, der skaliert werden soll.
`r1`	`int`	Der minimale Rohwert, der in `data` gespeichert ist.
`r2`	`int`	Der maximale Rohwert, der in `data` gespeichert ist.
`e1`	`int`	Der Konstruktionswert, der `r1` entspricht.
`e2`	`int`	Der Konstruktionswert, der `r2` entspricht.

Beschreibung

Diese Funktion skaliert das Argument `data` linear, wenn es Werte zwischen `r1` und `r2` enthält, und generiert einen Rückgabewert zwischen `e1` und `e2`. Die interne Mathematik wird mit 64-Bit-Ganzzahlen implementiert, wodurch Überläufe vermieden werden, die entstehen könnten, wenn versucht wird, mit den Crimson-eigenen mathematischen Operatoren sehr große Werte zu skalieren.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
Data = Scale([D100], 0, 4095, 0, 99999)
```

SendFile(*rcpt, file*)

ARGUMENT	TYP	BESCHREIBUNG
`rcpt`	`int`	Der Index des Empfängers im Adressbuch der Datenbank.
`file`	`cstring`	Der Pfad und der Dateiname, die gesendet werden sollen.

Beschreibung

Sendet eine E-Mail vom Crimson-Gerät mit der angegebenen Datei als Anhang. Die Funktion gibt sofort zurück, nachdem die erforderliche E-Mail der Mail-Warteschlange des Systems hinzugefügt wurde. Die Nachricht wird mit dem entsprechenden Mail-Transport gesendet, wie in der Datenbank konfiguriert.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
SendFile(0, "/LOGS/LOG1/260706.csv")
```

SendFileEx(*rcpt, file, subject, flag*)

ARGUMENT	TYP	BESCHREIBUNG
`rcpt`	`int`	Der Index des Empfängers im Adressbuch der Datenbank.
`file`	`cstring`	Der Pfad und der Dateiname, die gesendet werden sollen.
`subject`	`cstring`	Der Betreff der E-Mail.
`flag`	`int`	Nicht verwendet. Sollte auf Null gesetzt sein.

Beschreibung

Sendet eine E-Mail vom Crimson-Gerät mit der angegebenen Datei als Anhang sowie mit der angegebenen Betreffzeile. Die Funktion gibt sofort zurück, nachdem die erforderliche E-Mail der Mail-Warteschlange des Systems hinzugefügt wurde. Die Nachricht wird mit dem entsprechenden Mail-Transport gesendet, wie in der Datenbank konfiguriert.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
SendFileEx(0, "/LOGS/LOG1/260706.csv", "Test Email", 0)
```

SendMail(*rcpt, subject, body*)

ARGUMENT	TYP	BESCHREIBUNG
`rcpt`	`int`	Der Index des Empfängers im Adressbuch der Datenbank.
`subject`	`cstring`	Die erforderliche Betreffzeile für die E-Mail.
`body`	`cstring`	Der erforderliche Text der E-Mail.

Beschreibung

Sendet eine E-Mail vom Crimson-Gerät. Die Funktion gibt sofort zurück, nachdem die erforderliche E-Mail der Mail-Warteschlange des Systems hinzugefügt wurde. Die Nachricht wird mit dem entsprechenden Mail-Transport gesendet, wie in der Datenbank konfiguriert.

Hinweis: Der erste Empfänger ist 0.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
SendMail(1, "Test Subject Line", "Test Body Text")
```

Set(*tag, value*)

ARGUMENT	TYP	BESCHREIBUNG
`tag`	`int/float`	Das zu ändernde Tag.
`value`	`int/float`	Der zuzuweisende Wert.

Beschreibung

Mit dieser Funktion wird das angegebene Tag auf den angegebenen Wert gesetzt. Sie unterscheidet sich dadurch von dem normalerweise verwendeten Zuweisungsoperator, dass sie (a) alle in der Warteschlange befindlichen Schreibvorgänge für dieses Tag löscht und sie durch einen sofortigen Schreibvorgang des angegebenen Werts ersetzt. Sie wird daher in Situationen verwendet, in denen das normale Schreibverhalten von Crimson nicht erforderlich ist.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
Set(Tag1, 100)
```

SetIconLed(*id*, *state*)

ARGUMENT	TYP	BESCHREIBUNG
`id`	`int`	Die ID der Symbol-LED.
`state`	`int`	Der Status der Symbol-LED.

Beschreibung

Mit dieser Funktion wird der Status der angegebenen Symbol-LED auf den erforderlichen Status gesetzt:

Funktionstyp

ID	LED
1	`Alarm`
2	`Orb`
3	`Home`

Diese Funktion ist *passiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
SetIconLed(1, 0)
```

SetIntTag(*index*, *value*)

ARGUMENT	TYP	BESCHREIBUNG
`index`	`int`	Indexnummer des Tags.
`value`	`int`	Der zuzuweisende Wert.

Beschreibung

Mit dieser Funktion wird das von `index` angegebene Tag auf den angegebenen Wert gesetzt. Der Index kann mithilfe der Funktion `FindTagIndex()` aus der Tag-Bezeichnung abgerufen werden. Diese Funktion erfordert, dass das Zieltag eine ganze Zahl ist.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
SetIntTag(5,1234)
```

SetLanguage(*code*)

ARGUMENT	TYP	BESCHREIBUNG
`code`	`int`	Die auszuwählende Sprache.

Beschreibung

Setzt die aktuelle Sprache der Bedienoberfläche auf die durch `code` angegebene Sprache.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
SetLanguage(1)
```

SetNow(*time*)

ARGUMENT	TYP	BESCHREIBUNG
`time`	`int`	Die neu einzustellende Uhrzeit.

Beschreibung

Legt die aktuelle Uhrzeit über eine Ganzzahl fest, die die Anzahl der Sekunden darstellt, die seit dem 1. Januar 1997 verstrichen sind. Die Ganzzahl wird in der Regel über die anderen Zeit-/Datumsfunktionen generiert.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
SetNow(252288000)
```

SetRealTag(*index, value*)

ARGUMENT	TYP	BESCHREIBUNG
`index`	`int`	Die Indexnummer des Tags.
`value`	`float`	Der zuzuweisende Wert.

Beschreibung

Mit dieser Funktion wird das von `index` angegebene Tag auf den angegebenen Wert gesetzt. Der Index kann mithilfe der Funktion `FindTagIndex()` aus der Tag-Bezeichnung abgerufen werden. Diese Funktion funktioniert nur, wenn das ausgewählte Tag eine Gleitkommazahl ist.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
SetRealTag(5, 12.55)
```

Legt für das tatsächliche Tag des Index 5 den Wert 12.55 fest.

SetStringTag(*index, data*)

ARGUMENT	TYP	BESCHREIBUNG
`index`	`int`	Indexnummer des Tags.
`data`	`cstring`	Der zuzuweisende Wert.

Beschreibung

Mit dieser Funktion wird das von `index` angegebene Tag auf den angegebenen Wert gesetzt. Der Index kann mithilfe der Funktion `FindTagIndex()` aus der Tag-Bezeichnung abgerufen werden. Diese Funktion funktioniert nur, wenn das Zieltag eine Zeichenfolge ist.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Sgn(*value*)

ARGUMENT	TYP	BESCHREIBUNG
`value`	`int / float`	Der zu verarbeitende Wert.

Beschreibung

Gibt -1 zurück, wenn `value` kleiner als Null ist, +1, wenn der Wert größer als Null ist, oder 0, wenn er gleich Null ist.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int` oder `float`, abhängig vom Typ des Arguments für `value`.

Beispiel

```
State = Sgn(Level)+1
```

ShowMenu(*name*)

ARGUMENT	TYP	BESCHREIBUNG
`name`	`Bildschirmseite`	Die Anzeigeseite, die als Popup-Menü angezeigt werden soll.

Beschreibung

Zeigt die Seite an, die als Popup-Menü angegeben ist. Popup-Menüs werden über dem angezeigt, was bereits auf dem Bildschirm zu sehen ist, und sind an der linken Seite des Displays ausgerichtet.

Funktionstyp

Diese Funktion ist aktiv.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
ShowMenu(Page2)
```

ShowModal(*name*)

ARGUMENT	TYP	BESCHREIBUNG
`name`	`Bildschirmseite`	Die Seite, die als modales Popup angezeigt werden soll.

Beschreibung

Zeigt die Seite `name` als Popup auf dem Display des Crimson-Geräts an. Das Popup wird auf dem Display zentriert und über der vorhandenen Seite oder vorhandenen Popups angezeigt. Das Popup wird nicht entfernt, und die Funktion gibt erst dann zurück, wenn ein Aufruf an `EndModal()` erfolgt. An diesem Punkt wird der an diese Funktion übergebene Wert von `ShowModal()` zurückgegeben.

Modale Popups dienen zur Implementierung von Benutzeroberflächenfunktionen wie Bestätigungs-Popups in einem Programm. Beispielsweise soll der Benutzer bestätigen, dass eine bestimmte Datei tatsächlich gelöscht werden soll. Modale Popups machen dies einfacher und erfordern die Erstellung komplexer Zustandsautomaten.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
int
```

Beispiel

```
if( ShowModal(ConfirmDelete) == 1 ) {
DeleteFile(OpenFile("file.dat", 1));
}
```

ShowNested(*name*)

ARGUMENT	TYP	BESCHREIBUNG
`name`	`Bildschirmseite`	Die Seite, die als Popup angezeigt werden soll.

Beschreibung

Zeigt die Seite `name` als Popup auf dem Display des Crimson-Geräts an. Das Popup wird auf dem Display zentriert und über der vorhandenen Seite oder vorhandenen Popups angezeigt. Das Popup-Fenster kann entfernt werden, indem Sie entweder die Funktion `HidePopup()` oder `HideAllPopups()` aufrufen. Es wird auch vom Display entfernt, wenn durch Aufrufen der Funktion `GotoPage()` oder durch eine entsprechend definierte Tastaturaktion eine neue Seite ausgewählt wird.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

ShowPopup(*name*)

ARGUMENT	TYP	BESCHREIBUNG
`name`	`Bildschirmseite`	Die Seite, die als Popup angezeigt werden soll.

Beschreibung

Zeigt die Seite `name` als Popup auf dem Display des Crimson-Geräts an. Das Popup wird auf dem Display zentriert und über der vorhandenen Seite angezeigt. Das Popup kann durch Aufrufen der Funktion `HidePopup()` entfernt werden. Es wird auch vom Display entfernt, wenn durch Aufrufen der Funktion `GotoPage()` oder durch eine entsprechend definierte Tastaturaktion eine neue Seite ausgewählt wird.

Funktionstyp

Diese Funktion ist aktiv.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
ShowPopup(Popup1)
```

sin(*theta*)

ARGUMENT	TYP	BESCHREIBUNG
`theta`	`float`	Der zu verarbeitende Winkel, in Radianten.

Beschreibung

Gibt den Sinus des Winkels `theta` zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`float`

Beispiel

```
yp = radius*sin(theta)
```

sinR64(*result, tag*)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag`	`int`	Der zu verarbeitende Winkel, in Radianten.

Beschreibung

Berechnet den Sinus von *tag* unter Verwendung von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in *result*. Der Eingabeoperand *tag* sollte von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu AddR64 ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`void`

Beispiel

```
sinR64(result[0], tag[0])
```

SirenOn()

ARGUMENT	TYP	BESCHREIBUNG
`kein`		

Beschreibung

Schaltet die interne Sirene der Bedienoberfläche ein.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
SirenOn()
```

Sleep(*period*)

ARGUMENT	TYP	BESCHREIBUNG
`period`	`int`	Zeitraum des Ruhemodus in Millisekunden.

Beschreibung

Versetzt die aktuelle Aufgabe für die angegebene Anzahl von Millisekunden in den Ruhemodus. Diese Funktion wird normalerweise in Programmen verwendet, die im Hintergrund ausgeführt werden oder die benutzerdefinierte Kommunikation mit RAW-Port-Treibern implementieren. Es wird davon abgeraten, sie als Reaktion auf Auslöser oder Tastendruck aufzurufen.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
Sleep(100)
```

Sqrt(*value*)

ARGUMENT	TYP	BESCHREIBUNG
`value`	`int/float`	Der zu verarbeitende Wert.

Beschreibung

Gibt die Quadratwurzel von `value` zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int` oder `float`, abhängig vom Typ des Arguments für `value`.

Beispiel

```
Flow = Const * Sqrt(Input)
```

SqrtR64(result, tag)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag`	`int`	Der zu verarbeitende Wert.

Beschreibung

Berechnet die Quadratwurzel von *tag* unter Verwendung von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in *result*. Der Eingabeoperand *tag* sollte von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu `AddR64` ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`void`

Beispiel

```
SqrtR64(result[0], tag[0])
```

StdDev(*element, count*)

ARGUMENT	TYP	BESCHREIBUNG
`element`	`int/float`	Das erste Array-Element, das verarbeitet werden soll.
`count`	`int`	Die Anzahl der zu verarbeitenden Elemente.

Beschreibung

Gibt die Standardabweichung von `count` (Anzahl) Array-Elementen ab `element` zurück, davon ausgehend, dass die Datenpunkte eine Stichprobe der untersuchten Population darstellen. Wenn Sie die Standardabweichung der gesamten Population ermitteln müssen, verwenden Sie stattdessen die Funktion `PopDev`.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
float
```

Beispiel

```
Dev = StdDev(Data[0], 10)
```

StopSystem()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Stoppt das Crimson-Gerät, damit ein Benutzer die Datenbank aktualisieren kann. Diese Funktion wird normalerweise verwendet, wenn die serielle Programmierung in Bezug auf eine Einheit erforderlich ist, deren Programmierschnittstelle für die Kommunikation zugewiesen wurde. Durch das Aufrufen dieser Funktion wird die gesamte Kommunikation abgeschaltet, sodass der Port wieder als Programmierschnittstelle fungieren kann.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
StopSystem()
```

Strip(*text, target*)

ARGUMENT	TYP	BESCHREIBUNG
`text`	`cstring`	Die zu bearbeitende Zeichenfolge.
`target`	`int`	Das zu entfernende Zeichen.

Beschreibung

Entfernt alle Vorkommen eines bestimmten Zeichens aus einer Textzeichenfolge.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`cstring`

Beispiel

```
Text = Strip("Mississippi", 's')
```

SubR64(result, tag1, tag2)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag1`	`int`	Der Minuend-Wert.
`tag2`	`int`	Der Subtrahend-Wert.

Beschreibung

Berechnet den Wert von *tag1* minus *tag2* mithilfe von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in *result*. Die Eingabeoperanden *tag1* und *tag2* sollten von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu `AddR64` ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`void`

Beispiel

```
SubR64(result[0], tag1[0], tag2[0])
```

SubU32(tag1, tag2)

ARGUMENT	TYP	BESCHREIBUNG
`tag1`	`int`	Das Minuend-Tag.
`tag2`	`int`	Das Subtrahend-Tag.

Beschreibung

Gibt den Wert von *tag1* minus *tag2* in einem Kontext ohne Vorzeichen zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
Result = SubU32(tag1, tag2)
```

Sum(*element, count*)

ARGUMENT	TYP	BESCHREIBUNG
`element`	`int/float`	Das erste Array-Element, das verarbeitet werden soll.
`count`	`int`	Die Anzahl der zu verarbeitenden Elemente.

Beschreibung

Gibt die Summe der `count` (Anzahl) Array-Elemente ab `element` zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int` oder `float`, abhängig vom Typ des Arguments für `value`.

Beispiel

```
Total = Sum(Data[0], 10)
```

tan(*theta*)

ARGUMENT	TYP	BESCHREIBUNG
`theta`	`float`	Der zu verarbeitende Winkel, in Radianten.

Beschreibung

Gibt den Tangens des Winkels `theta` zurück.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`float`

Beispiel

```
yp = xp * tan(theta)
```

tanR64(result, tag)

ARGUMENT	TYP	BESCHREIBUNG
`result`	`int`	Das Ergebnis.
`tag`	`int`	Der zu verarbeitende Winkel, in Radianten.

Beschreibung

Berechnet den Tangens von *tag* unter Verwendung von 64-Bit-Gleitkommazahlen mit doppelter Genauigkeit und speichert das Ergebnis in *result*. Der Eingabeoperand tag sollte von einer der bereitgestellten 64-Bit-Konvertierungsfunktionen oder von einem Treiber abgerufen werden, der Werte mit doppelter Genauigkeit lesen kann. Alle Argumente für diese Funktion müssen ganzzahlige Arrays mit einer Länge von 2 sein. Im Eintrag zu `AddR64` ist ein detailliertes Beispiel angegeben.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`void`

Beispiel

```
tanR64(result[0], tag[0])
```

TestAccess(*rights, prompt*)

ARGUMENT	TYP	BESCHREIBUNG
`rights`	`int`	Die erforderlichen Zugriffsrechte.
`prompt`	`cstring`	Die Eingabeaufforderung, die im Anmelde-Popup verwendet werden soll.

Beschreibung

Gibt den Wert *true* oder *false* zurück, je nachdem, ob der aktuelle Benutzer Zugriffsrechte besitzt, die durch den Parameter `rights` definiert sind. Dieser Parameter enthält eine Bitmaske, um die verschiedenen benutzerdefinierten Rechte darzustellen, wobei das Bit 0 (d. h. das Bit mit dem Wert 0x01) für das Benutzerrecht 1 steht, Bit 1 (d. h. das Bit mit dem Wert 0x02) für das Benutzerrecht 2 und so weiter. Wenn aktuell kein Benutzer angemeldet ist, zeigt das System ein Popup-Fenster an, um nach Benutzeranmeldeinformationen zu fragen, und verwendet dazu das Argument `prompt`, um anzugeben, warum das Popup angezeigt wird. Die Funktion wird in der Regel in Programmen verwendet, die eine Reihe von Aktionen ausführen, die möglicherweise einer Sicherheitsstufe unterliegen und die andernfalls durch ein Anmeldefenster unterbrochen würden. Durch die Ausführung dieser Funktion vor dem Ausführen der Aktionen können Sie dem Benutzer einen besseren Hinweis geben, warum eine Anmeldung erforderlich ist, und Sie können einen Sicherheitsfehler nach Ausführung einer Reihe von Vorgängen vermeiden.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
if( TestAccess(1, "Clear all data?") ) {
Data1 = 0;
Data2 = 0;
Data3 = 0;
}
```

TextToAddr(*addr*)

ARGUMENT	TYP	BESCHREIBUNG
`addr`	`cstring`	Die Adresse in Dezimalschreibweise.

Beschreibung

Konvertiert eine Zeichenfolge in Dezimalschreibweise in eine 32-Bit-IP-Adresse.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiel

```
ip = TextToAddr("192.168.0.1")
```

TextToFloat(*string*)

ARGUMENT	TYP	BESCHREIBUNG
`string`	`cstring`	Die zu bearbeitende Zeichenfolge.

Beschreibung

Gibt den Wert von `string` zurück und behandelt die Zeichenfolge als Gleitkommazahl. Diese Funktion wird häufig zusammen mit `Mid` verwendet, um Werte aus Zeichenfolgen zu extrahieren, die von seriellen RAW-Ports empfangen wurden. Sie kann auch dazu verwendet werden, andere Zeichenfolgenwerte in Gleitkommazahlen zu konvertieren.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
float
```

Beispiel

```
Data = TextToFloat("3.142")
```

TextToInt(*string, radix*)

ARGUMENT	TYP	BESCHREIBUNG
`string`	`cstring`	Die zu bearbeitende Zeichenfolge.
`radix`	`int`	Die zu verwendende Zahlenbasis.

Beschreibung

Gibt den Wert von `string` zurück, wobei er als Basiswurzel `radix` behandelt wird. Diese Funktion wird häufig zusammen mit `Mid` verwendet, um Werte aus Zeichenfolgen zu extrahieren, die von seriellen RAW-Ports empfangen wurden. Sie kann auch dazu verwendet werden, andere Zeichenfolgenwerte in Ganzzahlen umzuwandeln.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

```
int
```

Beispiel

```
Data = TextToInt("1234", 10)
```

TextToR64(input, output)

ARGUMENT	TYP	BESCHREIBUNG
`input`	`cstring`	Der zu konvertierende Text.
`output`	`int`	Das Ziel des 64-Bit-Ergebnisses.

Beschreibung

Interpretiert den in der Zeichenfolge *input* gespeicherten Wert als 64-Bit-Gleitkommazahl mit doppelter Genauigkeit und speichert das Ergebnis als Array mit der Länge 2 in *output*. Das Tag *output* muss daher ein ganzzahliges Array mit einem Umfang von mindestens 2 sein. Nach der Ausführung dieser Funktion ist der in *result* gespeicherte Wert für die Verwendung in anderen mathematischen 64-Bit-Funktionen geeignet. Ein Beispiel für die beabsichtigte Verwendung dieser Funktion finden Sie im Eintrag zu `AddR64`.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`void`

Beispiel

```
TextToR64(input, output[0])
```

Time(*h*, *m*, *s*)

ARGUMENT	TYP	BESCHREIBUNG
`h`	`int`	Die zu codierende Stunde von 0 bis 23.
`m`	`int`	Die zu codierende Minute von 0 bis 59.
`s`	`int`	Die zu codierende Sekunde von 0 bis 59.

Beschreibung

Gibt einen Wert zurück, der die angegebene Zeit als die Anzahl der seit Mitternacht verstrichenen Sekunden darstellt. Dieser Wert kann dann mit anderen Zeit-/Datumsfunktionen verwendet werden. Er kann auch dem durch `Date` erzeugten Wert hinzugefügt werden, um einen Wert zu erzeugen, der auf eine bestimmte Uhrzeit und ein bestimmtes Datum verweist.

Funktionstyp

Diese Funktion ist *passiv*.

Rückgabetyp

`int`

Beispiel

```
t = Date(2000,12,31) + Time(12,30,0)
```

TxCAN(*port, data, id*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Der RAW-CAN-Port.
`data`	`int`	Das erste Array-Element, das Daten zur Übertragung enthält.
`id`	`int`	29-Bit CAN-Kennung.

Beschreibung

Sendet CAN-Nachrichten an einen Port, der mit `TxCANInit` initialisiert wurde. Die ersten vier Bytes der zu übertragenden Nachricht sollten mit Big Endian-Byteanordnung im ersten Element des Arrays gespeichert werden, wobei weitere Bytes in den nachfolgenden Array-Einträgen im gleichen Format folgen. Die Funktion gibt bei Erfolg einen Wert von 1 zurück.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
int
```

Beispiel

```
TxCAN(8, Data, 0x12345677)
```

TxCANInit(*port, id, dlc*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Der RAW-CAN-Port.
`id`	`int`	29-Bit CAN-Kennung.
`dlc`	`int`	Datenlänge 1 bis 8 Byte.

Beschreibung

Initialisiert CAN-Nachrichten, die über die CAN-Optionskarte gesendet werden sollen. Diese Funktion gibt bei Erfolg einen Wert von 1 oder bei einem Fehler einen Wert von 0 zurück. Aufrufe sollten erst nach dem Start des Systems erfolgen und jede 29-Bit-Kennung darf nur einmal initialisiert werden.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

Beispiel

```
TxCANInit(8, 0x12345677, 8)
```

UseCameraSetup(*port*, *camera*, *index*)

ARGUMENT	TYP	BESCHREIBUNG
`port`	`int`	Die Nummer des Ports, an den die Kamera angeschlossen ist.
`camera`	`int`	Die Gerätenummer der Kamera.
`index`	`int`	Die Prüfdateinummer in der Kamera.

Beschreibung

Diese Funktion wählt die Prüfdatei aus, die von der Kamera verwendet werden soll. Die Zahl, die im Argument `port` platziert werden soll, ist die Portnummer, an die der Treiber gebunden ist. Das Argument `camera` ist die Gerätenummer, die in der Statusleiste von Crimson angezeigt wird, wenn die Kamera ausgewählt ist. Unter einem Treiber kann mehr als eine Kamera angeschlossen werden. Der `index` steht für die Prüfdateinummer in der Kamera. Die Funktion gibt bei Erfolg “true” zurück, andernfalls “false”. Diese Funktion sollte in einem Benutzerprogramm aufgerufen werden, das im Hintergrund ausgeführt wird. Wird es im Vordergrund aufgerufen, wird die Benutzeroberfläche pausiert.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
int
```

Beispiel

```
Success = UseCameraSetup(4, 0, 1)
```

UserLogOff()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Bewirkt, dass der aktuelle Benutzer vom System abgemeldet wird. Alle künftigen Aktionen, für die Sicherheitszugriffsrechte erforderlich sind, führen zur Anzeige des Anmeldefensters, um die Eingabe von Anmeldedaten zu ermöglichen.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
UserLogOff()
```

UserLogOn()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Erzwingt die Anzeige des Anmeldefensters, um die Eingabe von Benutzeranmeldedaten zu ermöglichen. Sie müssen diese Funktion normalerweise nicht verwenden, da Crimson 3.1 zur Eingabe von Anmeldedaten auffordert, wenn eine Aktion durchgeführt wird, für die eine Sicherheitsfreigabe erforderlich ist.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

Beispiel

```
UserLogOn()
```

WaitData(*data, count, time*)

ARGUMENT	TYP	BESCHREIBUNG
`data`	`any`	Das erste Array-Element, das gelesen werden soll.
`count`	`int`	Die Anzahl der zu lesenden Elemente.
`time`	`int`	Die Timeout-Zeit in Millisekunden.

Beschreibung

Fordert, dass bei der nächsten Kommunikationsprüfung `count` (Anzahl) Elemente ab dem Array-Element `data` gelesen werden. Diese Funktion wird bei Arrays verwendet, die externen Daten zugeordnet wurden und deren Leserichtlinie auf *manuell lesen* gesetzt wurde. Im Gegensatz zu `ReadData()` wartet diese Funktion bis zur im Parameter `time` angegebenen Zeitdauer, damit die Daten gelesen werden können. Der Rückgabewert ist 1, wenn der Lesevorgang innerhalb dieses Zeitraums abgeschlossen ist, andernfalls 0.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

Beispiel

```
status = WaitData(array1[8], 10, 1000)
```

WriteAll()

ARGUMENT	TYP	BESCHREIBUNG
kein		

Beschreibung

Erzwingt das Schreiben aller zugeordneten Tags, die nicht schreibgeschützt sind, auf ihre externen Geräte.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

Diese Funktion gibt keinen Wert zurück.

WriteFile(*file, text*)

ARGUMENT	TYP	BESCHREIBUNG
`file`	`int`	Das Datei-Handle, wie es von OpenFile gefordert wird.
`Text`	`cstring`	Der Text, der in die Datei geschrieben werden soll.

Beschreibung

Schreibt eine Zeichenfolge mit einer Länge von bis zu 512 Zeichen in die angegebene Datei und gibt die Anzahl der erfolgreich geschriebenen Bytes zurück. Diese Funktion beinhaltet nicht automatisch einen Zeilenvorschub und Wagenrücklauf am Ende. Weitere Informationen zur Programmierung finden Sie unter `WriteFileLine()`.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

`int`

Beispiel

```
count = WriteFile(hFile, "Writing text to file.")
```

WriteFileLine(*file, text*)

ARGUMENT	TYPE	BESCHREIBUNG
`file`	`int`	Datei-Handle, wie es von OpenFile benötigt wird.
`text`	`cstring`	Text, der in die Datei geschrieben werden soll.

Beschreibung

Schreibt eine Zeichenfolge in die angegebene Datei und gibt die Anzahl der erfolgreich geschriebenen Bytes zurück, einschließlich der Wagenrücklauf- und Zeilenvorschubzeichen, die an jede Zeile angefügt werden.

Funktionstyp

Diese Funktion ist *aktiv*.

Rückgabetyp

```
int
```

Beispiel

```
count = WriteFileLine(hFile, "Writing text to file.")
```

Kapitel 3 Systemvariablen

In diesem Kapitel werden die in Crimson 3.1 verfügbaren Systemvariablen beschrieben. Diese Systemvariablen können innerhalb von Aktionen oder Ausdrücken aufgerufen werden, wie im Crimson 3.1 User Manual beschrieben. Systemvariablen werden entweder verwendet, um den Status des Systems wiederzugeben oder das Verhalten des Systems auf irgendeine Weise zu verändern. Wenn der Systemstatus wiedergegeben wird, sind Systemvariablen schreibgeschützt. Wird eine Systemvariable zum Ändern des Systemverhaltens verwendet, kann ihr ein Lese-/Schreibwert zugewiesen werden.

ActiveAlarms

Beschreibung

Gibt die Anzahl der aktuell aktiven Alarme zurück.

Variablentyp

`int`

Zugriffstyp

Schreibgeschützt

CommsError

Beschreibung

Gibt eine Bitmaske zurück, die angibt, ob das jeweilige Kommunikationsgerät offline ist. Ein Wert von 1 an einer bestimmten Bit-Position zeigt an, dass im entsprechenden Gerät Kommunikationsfehler vorliegen. Bit 0 (d. h. das Bit mit einem Wert von 1) entspricht dem ersten Kommunikationsgerät.

Variablentyp

`int`

Zugriffstyp

Schreibgeschützt

DispBrightness

Beschreibung

Gibt eine Zahl zurück, die die Helligkeit der Anzeige von 0 bis 100 angibt, wobei der Wert Null für “Aus” steht.

Variablentyp

`int`

Zugriffstyp

Lesen / Schreiben

DispContrast

Beschreibung

Gibt eine Zahl zurück, die den Anzeigekontrast von 0 bis 100 angibt.

Variablentyp

`int`

Zugriffstyp

Lesen / Schreiben

DispCount

Beschreibung

Gibt eine Zahl zurück, die die Anzahl der Anzeigeaktualisierungen seit dem letzten Zurücksetzen angibt.

Variablentyp

`int`

Zugriffstyp

Schreibgeschützt

DispUpdates

Beschreibung

Gibt eine Zahl zurück, die angibt, wie oft die Anzeige pro Sekunde aktualisiert wird.

Variablentyp

`int`

Zugriffstyp

Schreibgeschützt

IconBrightness

Beschreibung

Enthält einen Wert, der die Helligkeit der Symbol-LEDs von 0 bis 100 angibt, wobei Null für “Aus” steht.

Variablentyp

`int`

Zugriffstyp

Lesen / Schreiben

IsPressed

Beschreibung

Gibt “true” zurück, wenn das aktuelle Primitiv über den Touchscreen oder Webserver angeklickt wird, andernfalls “false”. Die Variable ist nur innerhalb von Ausdrücken oder Aktionen gültig, die innerhalb der Konfiguration des Primitivs oder in Vordergrundprogrammen vorhanden sind, die von diesen Orten aufgerufen werden. Wird in anderen Situation darauf verwiesen, wird ein nicht definierter Wert erzeugt.

Variablentyp

`int`

Zugriffstyp

Schreibgeschützt

IsSirenOn

Beschreibung

Gibt “true” zurück, wenn der akustische Signalgeber der Frontplatte eingeschaltet ist, andernfalls “false”.

Variablentyp

`int`

Zugriffstyp

Schreibgeschützt

Pi

Beschreibung

Gibt *pi* als Gleitkommazahl zurück.

Variablentyp

`float`

Zugriffstyp

Schreibgeschützt

TimeNow

Beschreibung

Gibt die aktuelle Uhrzeit und das aktuelle Datum als Anzahl der Sekunden seit dem Bezugspunkt vom 1. Januar 1997 zurück. Dieser Wert kann dann mit anderen Zeit-/Datumsfunktionen verwendet werden. Durch das Schreiben in diese Variable wird die Echtzeituhr auf die entsprechende Uhrzeit eingestellt.

Variablentyp

`int`

Zugriffstyp

Lesen / Schreiben

TimeZone

Beschreibung

Gibt die Zeitzone in Stunden von -12 bis +12 zurück. Mit dem Befehl “Link Send Time” (Sendezeit verknüpfen) in Crimson werden die Zeit und die Zeitzone des Geräts auf die Werte des Computers gesetzt. Die spätere Änderung der Zeitzone stellt die Gerätezeit vor oder zurück. Beachten Sie, dass die Zeitzone nur angezeigt oder geändert werden kann, wenn der Zeitmanager aktiviert ist.

Variablentyp

`int`

Zugriffstyp

Lesen / Schreiben

TimeZoneMins

Beschreibung

Gibt die Zeitzone in Minuten von -720 bis +720 zurück. Mit dem Befehl “Link Send Time” (Sendezeit verknüpfen) in Crimson werden die Zeit und die Zeitzone des Geräts auf die Werte des Computers gesetzt. Die spätere Änderung der Zeitzone stellt die Gerätezeit vor oder zurück. Beachten Sie, dass “TimeZoneMins” nur angezeigt oder geändert werden kann, wenn der Zeitmanager aktiviert ist.

Variablentyp

```
int
```

Zugriffstyp

Lesen / Schreiben

Unaccepted Alarms

Beschreibung

Gibt die Anzahl der nicht angenommenen Alarme im System zurück.

Variablentyp

`int`

Zugriffstyp

Lesen

UnacceptedAndAutoAlarms

Beschreibung

Gibt die Anzahl der nicht angenommenen Alarme zurück, zusätzlich dazu die Anzahl der aktuell aktiven und für automatische Annahme konfigurierten Alarme.

Variablentyp

`int`

Zugriffstyp

Schreibgeschützt

UseDST

Beschreibung

Gibt den Sommerzeit-Status des Geräts zurück. Diese Variable fügt eine Stunde zur Gerätezeit hinzu, wenn sie auf “true” gesetzt ist. Beachten Sie, dass “UseDST” nur angezeigt oder geändert werden kann, wenn der Zeitmanager aktiviert ist.

Variablentyp

`flag`

Zugriffstyp

Lesen / Schreiben