DOKUMENTATION ISG-kernel

Funktionsbeschreibung Echtzeit-Zyklen

Kurzbezeichnung:
FCT-C32

www.isg-stuttgart.de
support@isg-stuttgart.de

Dokumentation Version: 1.12
12.12.2023

Vorwort

Rechtliche Hinweise

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte und der Funktionsumfang werden jedoch ständig weiterentwickelt. Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.

Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen, der zugehörigen Dokumentation und der Aufgabenstellung vertraut ist.

Zur Installation und Inbetriebnahme ist die Beachtung der Dokumentation, der nachfolgenden Hinweise und Erklärungen unbedingt notwendig. Das Fachpersonal ist verpflichtet, für jede Installation und Inbetriebnahme die zum betreffenden Zeitpunkt veröffentlichte Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbarer Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Weiterführende Informationen

Unter den Links (DE)

https://www.isg-stuttgart.de/produkte/softwareprodukte/isg-kernel/dokumente-und-downloads

bzw. (EN)

https://www.isg-stuttgart.de/en/products/softwareproducts/isg-kernel/documents-and-downloads

finden Sie neben der aktuellen Dokumentation weiterführende Informationen zu Meldungen aus dem NC-Kern, Onlinehilfen, SPS-Bibliotheken, Tools usw.

Haftungsausschluss

Änderungen der Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig.

Marken und Patente

Der Name ISG®, ISG kernel®, ISG virtuos®, ISG dirigent® und entsprechende Logos sind eingetragene und lizenzierte Marken der ISG Industrielle Steuerungstechnik GmbH.

Die Verwendung anderer in dieser Dokumentation enthaltene Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Copyright

Allgemeine- und Sicherheitshinweise

Verwendete Symbole und ihre Bedeutung

In der vorliegenden Dokumentation werden die folgenden Symbole mit nebenstehendem Sicherheitshinweis und Text verwendet. Die (Sicherheits-) Hinweise sind aufmerksam zu lesen und unbedingt zu befolgen!

Symbole im Erklärtext

➢ Gibt eine Aktion an.

⇨ Gibt eine Handlungsanweisung an.

⚠ GEFAHR

Akute Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht unmittelbare Gefahr für Leben und Gesundheit von Personen!

⚠ VORSICHT

Schädigung von Personen und Maschinen!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, können Personen und Maschinen geschädigt werden!

Achtung

Einschränkung oder Fehler

Dieses Symbol beschreibt Einschränkungen oder warnt vor Fehlern.

Hinweis

Tipps und weitere Hinweise

Dieses Symbol kennzeichnet Informationen, die zum grundsätzlichen Verständnis beitragen oder zusätzliche Hinweise geben.

Beispiel

Allgemeines Beispiel

Beispiel zu einem erklärten Sachverhalt.

Programmierbeispiel

NC-Programmierbeispiel

Programmierbeispiel (komplettes NC-Programm oder Programmsequenz) der beschriebenen Funktionalität bzw. des entsprechenden NC-Befehls.

Versionshinweis

Spezifischer Versionshinweis

Optionale, ggf. auch eingeschränkte Funktionalität. Die Verfügbarkeit dieser Funktionalität ist von der Konfiguration und dem Versionsumfang abhängig.

Inhaltsverzeichnis

1 Übersicht

Aufgabe

Echtzeit-Zyklen sind Zyklen, die im Echtzeit-Teil der Steuerung nebenläufig ausgeführt werden. Dadurch ist es möglich, auf Echtzeit-Einflüsse zu reagieren.
Mögliche Reaktionen sind:

- Spindelsteuerung
- Einzelachsbewegungen
- Ausgabe von M-Funktionen
- Ausführung von NC-Zusatzfunktionen über #-Befehle

Hinweis

Diese Funktionalität ist eine lizenzpflichtige Zusatzoption.

Versionshinweis

Diese Funktionalität ist verfügbar ab CNC-Version V3.1.3105

Parametrierung

Die Funktionalität muss über den Kanalparameter P-CHAN-00406 [▸ 41] aktiviert werden.

Zusätzliche Einstellmöglichkeiten des Anwenders:

- Verfügbarer Speicher (P-CHAN-00407) der Echtzeit-Zyklen
- Optionale Überwachung der Ausführungsdauer der Echtzeit-Zyklen pro CNC-Takt, um Echtzeit-Überschreitungen zu verhindern (P-CHAN-00425, P-CHAN-00426 [▸ 42] und P-CHAN-00427 [▸ 43])

Programmierung

Ein Echtzeit-Zyklus wird im NC-Programm als abgeschlossener Bereich über den NC-Befehl **#RT CYCLE** definiert.

Obligatorischer Hinweis zu Verweisen auf andere Dokumente

Zwecks Übersichtlichkeit wird eine verkürzte Darstellung der Verweise (Links) auf andere Dokumente bzw. Parameter gewählt, z.B. [PROG] für Programmieranleitung oder P-AXIS-00001 für einen Achsparameter.

Technisch bedingt funktionieren diese Verweise nur in der Online-Hilfe (HTML5, CHM), allerdings nicht in PDF-Dateien, da PDF keine dokumentenübergreifenden Verlinkungen unterstützt.

2 Echtzeit-Zyklus

Umfang eines Echtzeit-Zyklus

Ein Echtzeit-Zyklus wird in einem NC-Programm definiert. Er besteht aus einer Reihe von Anweisungen und Kontrollstrukturen, wie z.B. **$IF-Bedingungen**.

Nach dem Laden und Starten in den Echtzeit-Teil der Steuerung wird der Echtzeit-Zyklus in jedem CNC-Takt **komplett** durchlaufen.

2.1 Definition

Die Definition erfolgt in einem Block, der mit **#RT CYCLE** und **#RT CYCLE END** umschlossen ist:

```
Syntax:
#RT CYCLE [DEF] [[ ID =.. ] SCOPE =.. END_ACTION =..]
;…Anweisungen
#RT CYCLE END
```

DEF	Echtzeit-Zyklus wird nur definiert, aber noch nicht gestartet.
ID=..	Eindeutiger Identifier des Echtzeit-Zyklus, siehe Hinweis.
SCOPE	Gültigkeit, siehe Gültigkeit [▸ 9].
	Zulässige Kennungen sind **BLOCK**, **PROG** oder **GLOBAL**.
END_ACTION	Verhalten beim Beenden des Echtzeit-Zyklus, siehe Aktion beim Beenden [▸ 11].
	Zulässige Kennungen sind **MOVE_ABORT** oder **MOVE_CONT**.

Hinweis

Die ID des Echtzeit-Zyklus muss zwingend angegeben werden, wenn das Schlüsselwort DEF verwendet wird oder SCOPE = GLOBAL ist.

Ist die ID in diesen Fällen nicht angegeben, dann wird der Fehler ID 22003 ausgegeben.

2.2 Trennung von Definition und Aktivierung

Wenn das Schlüsselwort **DEF nicht** verwendet wird, dann wird der Echtzeit-Zyklus definiert und **sofort gestartet** wenn er in den Echtzeit-Teil der Steuerung geladen wurde.

Programmierbeispiel

Echtzeit-Zyklus definieren und starten

```
; Echtzeit-Zyklus definieren und starten
#RT CYCLE [SCOPE = PROG]
  ; …
#RT CYCLE END

; Echtzeit-Zyklus ist bei diesem Satz schon aktiv
N10 G00 X100
```

Mit **DEF** wird der Echtzeit-Zyklus definiert, aber **noch nicht gestartet**. In diesem Fall muss zwingend eine ID programmiert werden. Die Aktivierung kann später mit **#RT CYCLE START** erfolgen, siehe Verwalten der Echtzeitzyklen [▸ 12].

Programmierbeispiel

Echtzeit-Zyklus nur definieren, aber noch nicht starten

```
; Echtzeit-Zyklus nur definieren, aber noch nicht starten
#RT CYCLE DEF [ID = 17 SCOPE = PROG]
  ; ...
#RT CYCLE END

; Echtzeit-Zyklus zu einem beliebigen Zeitpunkt starten
#RT CYCLE START [ID = 17]
```

2.3 Gültigkeit

Jeder Echtzeit-Zyklus hat eine definierte Gültigkeit, die seine Lebensdauer bestimmt. Sie wird über das Schlüsselwort **SCOPE** programmiert.

SCOPE	Gültigkeit
BLOCK	Gültig im nächsten NC-Satz.
PROG	Gültig für die Dauer des aktiven Hauptprogramms.
GLOBAL	Wirkt über das Ende des Hauptprogramms hinaus.

2.3.1 Gültigkeit BLOCK

Ein Echtzeit-Zyklus mit der Gültigkeit BLOCK wirkt für die Dauer des nächsten NC-Satzes.

Programmierbeispiel

Gültigkeit BLOCK

```
; Echtzeit-Zyklus definieren und starten
#RT CYCLE [SCOPE = BLOCK]
  ;...
#RT CYCLE END

; Echtzeit-Zyklus wirkt nur bei diesem NC-Satz
G00 X100
```

Wird der Echtzeit-Zyklus mit der Gültigkeit BLOCK zusätzlich mit dem Schlüsselwort DEF definiert, so kann dieser mehrfach verwendet werden.

Programmierbeispiel

Gültigkeit BLOCK - mehrfache Verwendung

```
; Echtzeit-Zyklus definieren, noch nicht starten
#RT CYCLE DEF [ID = 17 SCOPE = BLOCK]
  ; ...
#RT CYCLE END

; Echtzeit-Zyklus für den folgenden Satz verwenden
#RT CYCLE START [ID = 17]
G00 X100
; ...

; Echtzeit-Zyklus erneut verwenden
#RT CYCLE START [ID = 17]
G00 X200
;…

M30
```

2.3.2 Gültigkeit PROG

Ein Echtzeit-Zyklus mit der Gültigkeit PROG wirkt für die Dauer aktiven Hauptprogramms. Am Ende des Hauptprogramms (M30) wird er gelöscht.

Programmierbeispiel

Gültigkeit - PROG

```
; Echtzeit-Zyklus definieren und starten
#RT CYCLE [SCOPE = PROG]
;...
#RT CYCLE END

; Echtzeit-Zyklus wirkt ab hier
;...

M30
; Echtzeit-Zyklus wird hier gelöscht
```

2.3.3 Gültigkeit GLOBAL

Ein Echtzeit-Zyklus mit der Gültigkeit **GLOBAL** wirkt über das Ende des Hauptprogramms (M30) hinaus. In diesem Fall muss zwingend eine ID angegeben werden, um den Echtzeit-Zyklus später noch verwalten zu können.

Achtung

Während des Resets werden Zyklen mit Gültigkeit GLOBAL nicht durchlaufen.

2.4 Aktion beim Beenden

Ein Echtzeit-Zyklus kann auf zwei Arten beendet werden:

1. Er verlässt seine Gültigkeit.
2. Er wird durch **#RT CYCLE DELETE** explizit gelöscht.

Mit dem Schlüsselwort **END_ACTION** kann gesteuert werden, was mit Achsbewegungen geschehen soll, die der Echtzeit-Zyklus angestoßen hat und die beim Beenden noch in Ausführung sind.

END_ACTION	Bedeutung
MOVE_ABORT	Achsbewegungen abbrechen (Standard)
MOVE_CONT	Achsbewegungen fortsetzen

2.5 Zeitkritische Ausführung

Achtung

Jeder aktive Echtzeit-Zyklus wird in jedem CNC-Interpolatortakt ausgeführt. Durch die Programmierung von vielen Anweisungen ist es daher möglich, dass die Ausführung des Echtzeit-Zyklus zu viel Zeit des CNC-Taktes verbraucht. Mit P-CHAN-00425 ist es möglich, die Ausführungsdauer der Echtzeit-Zyklen zu begrenzen.

Die Parameter P-CHAN-00425, P-CHAN-00426 und P-CHAN-00427 stellen einen Sicherheitsmechanismus dar, um diese Echtzeit-Überläufe möglichst früh zu vermeiden.

2.6 Verwaltung der Echtzeit-Zyklen

Das Verhalten eines Echtzeit-Zyklus kann über folgende Schlüsselwörter beeinflusst werden:

#RT CYCLE [START | HOLD | CONTINUE | ABORT ACTION | DELETE] [ID=..]

Schlüsselwort	Beschreibung
START	Echtzeit-Zyklus wird jetzt durchlaufen, Zustand der **$IF**-Bedingung wird neu initialisiert, Bewegungen/Aktionen werden ausgeführt
HOLD	Echtzeit-Zyklus wird nicht mehr durchlaufen, Zustand der **$IF**-Bedingung bleibt erhalten, Bewegungen/Aktionen werden angehalten
CONTINUE	Mit **HOLD** angehaltener Echtzeit-Zyklus wird wieder durchlaufen, Zustand der **$IF**-Bedingung wie vor dem **HOLD**, zuvor angehaltene Bewegungen/Aktionen werden fortgesetzt
ABORT ACTION	Echtzeit-Zyklus wird weiter durchlaufen, Zustand der **$IF**-Bedingung bleibt erhalten, Bewegungen/Aktionen werden abgebrochen
DELETE	Echtzeit-Zyklus wird gelöscht

Programmierbeispiel

Verwaltung eines Echtzeit-Zyklus

```
; X-Achse auf 0mm fahren
G00 X0

; Echtzeit-Zyklus definieren, noch nicht starten
#RT CYCLE DEF [ID = 17 SCOPE = PROG]

  ; ACS-Position der X-Achse abfragen
  $IF ONCE V.RTA.ACS.ACT_POS.X > 200

    ; unabhaengige Z-Achsbewegung starten
    ; ...

  $ENDIF

#RT CYCLE END

; X-Achse verfahren
G00 X50

; Echtzeit-Zyklus starten
#RT CYCLE START [ID = 17]

; X-Achse verfahren, Z-Bewegung wird gestartet
G00 X250

; Z-Bewegung anhalten
#RT CYCLE HOLD [ID = 17]

; ...

; Z-Bewegung fortsetzen
#RT CYCLE CONTINUE [ID = 17]

; ...

; Hauptprogramm beenden
; PROG-Zyklus 17 wird automatisch gelöscht
M30
```

Programmierbeispiel

Globalen Echtzeit-Zyklus löschen

```
; Globalen Echtzeit-Zyklus definieren
#RT CYCLE [ID = 17 SCOPE = GLOBAL]

  ; Anweisungen
  ; ...

#RT CYCLE END

; ...

; Globalen Echtzeit-Zyklus explizit löschen
#RT CYCLE DELETE [ID = 17]

; Hauptprogramm beenden
M30
```

2.7 $IF-Kontrollstruktur

In Echtzeit-Zyklen wird zwischen

- $IF ohne Häufigkeit und
- $IF mit Häufigkeit unterschieden.

$IF-Bedingungen ohne Häufigkeit können zusätzlich mit **$ELSEIF** und **$ELSE** ergänzt werden. Dies entfällt bei $IF-Bedingungen mit Häufigkeitsangaben.

2.7.1 $IF ohne Häufigkeit

Der Anweisungsblock wird genau dann ausgeführt, wenn die Bedingung wahr ist.

Programmierbeispiel

$IF mit Mehrfachverzweigung

```
; Echtzeit-Zyklus definieren
#RT CYCLE [SCOPE = PROG]

  ; Wert der externen Variable abfragen
  $IF V.E.VALUE > 0
    ; Positiv-Zähler hochzählen
    V.E.COUNT_POS += 1
  $ELSEIF V.E.VALUE < 0
    ; Negativ-Zähler hochzählen
    V.E.COUNT_NEG += 1
  $ELSE
    ; Null-Zähler hochzählen
    V.E.COUNT_NULL += 1
  $ENDIF

#RT CYCLE END
```

2.7.2 $IF mit Häufigkeit

Die $IF-Kontrollstruktur hat im Echtzeit-Zyklus eine erweiterte Syntax. Jedes **$IF** kann durch eines von mehreren Schlüsselwörtern qualifiziert werden.

$IF [ONCE | EDGE | ALWAYS]=..

; Anweisungen

$ENDIF

Schlüsselwort	Bedeutung
ONCE	Wenn die Bedingung erfüllt ist, wird der Anweisungsteil genau ein einziges Mal ausgeführt.
EDGE	Bei jeder Änderung des Wertes der Bedingung von FALSE nach TRUE.
ALWAYS	Sobald die Bedingung einmal erfüllt ist, werden die Anweisungen in jedem Durchlauf ausgeführt.

Hinweis

Im Fall eines **$IF** mit Häufigkeitsangabe stehen die Mehrfachverzweigungen mittels **$ELSEIF** und **$ELSE** nicht zur Verfügung.

2.7.2.1 $IF ONCE

Wenn die Bedingung erstmalig erfüllt ist, dann wird der Anweisungsteil einmalig ausgeführt. Der Anweisungsteil wird in späteren Durchläufen nicht mehr ausgeführt, auch wenn die Bedingung erfüllt ist.

Programmierbeispiel

$IF ONCE

```
; Echtzeit-Zyklus definieren
#RT CYCLE [SCOPE = PROG]

  ; ACS-Position der X-Achse abfragen
  $IF ONCE V.RTA.ACS.ACT_POS.X > 200

    ; M-Funktion ausgeben
    M100

  $ENDIF

#RT CYCLE END
```

2.7.2.2 $IF EDGE

Bei jedem Zustandsübergang der Bedingung von **FALSE** nach **TRUE** (steigende Flanke) wird der Anweisungsblock einmal ausgeführt.

Programmierbeispiel

$IF EDGE

```
; Echtzeit-Zyklus definieren
#RT CYCLE [SCOPE = PROG]

  ; ACS-Position der X-Achse abfragen
  $IF EDGE V.RTA.ACS.ACT_POS.X > 200

    ; M-Funktion jedesmal ausgeben, wenn X die 200mm-Grenze
    ; in positiver Richtung überschreitet
    M100

  $ENDIF

#RT CYCLE END
```

2.7.2.3 $IF ALWAYS

Nachdem die Bedingung einmal erfüllt wurde, wird der Anweisungsblock zyklisch in jedem Takt ausgeführt, solange der Echtzeit-Zyklus aktiv ist.

Ab dem Zeitpunkt, ab dem die Bedingung erstmals erfüllt war, wird der Anweisungsteil in jedem Durchlauf ausgeführt. Die Bedingung wird in den nachfolgenden Durchläufen nicht mehr geprüft, sie wird als TRUE angenommen, solange de Echtzeit-Zyklus aktiv ist.

Programmierbeispiel

$IF ALWAYS

```
; Echtzeit-Zyklus definieren
#RT CYCLE [SCOPE = PROG]

  ; ACS-Position der X-Achse abfragen
  $IF ALWAYS V.RTA.ACS.ACT_POS.X > 200

    ; Sobald X die 200mm-Grenze überschreitet, wird die
    ; M-Funktion in jedem CNC-Takt ausgegeben.
    ; Auch wenn die X-Position wieder kleiner wird, wird
    ; M100 weiter ausgegeben.
    M100

  $ENDIF

#RT CYCLE END
```

2.7.2.4 Kombination $IF-Bedingungen mit und ohne Häufigkeit

Hinweis

$IF-Bedingungen mit und ohne Häufigkeitsangabe können ineinander geschachtelt werden. Dabei ist aber zu beachten, dass Anweisungen nur ausgeführt werden, wenn sie auch erreicht werden.

Beispielsweise führt ein einmaliges Aktivieren eines **ALWAYS**-Blocks nicht dazu, dass seine Anweisungen in allen folgenden CNC-Takten ausgeführt werden. Sie können durch eine übergeordnete **$IF-**Bedingung verhindert werden, falls diese ein negatives Ergebnis hat.

Wird im folgenden NC-Programm V.E.CONDITION wahr und anschließend V.E.VALUE > 100, wird der ALWAYS-Block durchlaufen. Dies gilt allerdings nur solange V.E.CONDITION den Wert 1 hat.

Programmierbeispiel

Geschachtelte $IF-Bedingungen

```
; Echtzeit-Zyklus definieren
#RT CYCLE [SCOPE = PROG]

  ; externe Bedingung
  $IF V.E.CONDITION == 1

    ; ALWAYS-Block
    $IF ALWAYS V.E.VALUE > 100

      ; ALWAYS-Anweisungen
      ; ...

    $ENDIF

  $ENDIF

#RT CYCLE END
; ...
; Hauptprogramm beenden
M30
```

2.8 Mathematische Ausdrücke

Im Echtzeit-Zyklus können mathematische Ausdrücke, Funktionen und Operatoren verwendet werden. Die gilt für die Bedingung einer **$IF**-Klausel, für Variablen-Zuweisungen etc.

Programmierbeispiel

Beispiele mathematischer Ausdrücke im Echtzeit-Zyklus

```
; Echtzeit-Zyklus definieren
#RT CYCLE [SCOPE = PROG]

  ; Ausdruck in Bedingung
  $IF SQRT[7 * 5] < 12 * SIN[30]

    ; ...

  $ENDIF

  ; Wert eines math. Ausdruck an externe Variable zuweisen
  V.E.VALUE1 = 12 * SIN[V.E.VALUE2] + SQRT[33]

#RT CYCLE END

; Hauptprogramm beenden
M30
```

3 Verwendbare Variablen

Innerhalb eines Echtzeit-Zyklus müssen synchrone Variablen zum Schreiben verwendet werden. Grund hierfür ist, dass die Werte dieser Variablen zum Ausführungszeitpunkt zur Verfügung stehen. Variablen, die nur zum Deklarationszeitpunkt zur Verfügung stehen, werden als Konstanten behandelt und sind daher nicht zuweisbar.

Folgende Variablen sind verwendbar:

- V.E-Variablen, wenn sie als synchron markiert sind
- V.RTG/V.RTA-Variablen
- V.CH-Variablen, wenn sie als synchron markiert sind

Auf diese Variablenarten wird in den nächsten Unterkapiteln eingegangen.

Hinweis

Nur synchrone Variablen dürfen innerhalb eines Echtzeit-Zyklus geschrieben werden.

Werden asynchrone Variablen geschrieben, so wird der Fehler ID 22009 ausgegeben.

3.1 Externe Variablen

Aus dem Echtzeit-Zyklus sind Lese-/Schreibzugriffe auf kanalspezifische und globale externe Variablen **V.E.*** möglich. Für einen Schreibzugriff muss die Variable als synchron definiert sein.

Programmierbeispiel

Definition einer synchronisierten V.E.-Variablen

```
...
var[0].name                    VALUE1
var[0].type                    REAL64
var[0].scope                   GLOBAL
var[0].synchronisation         TRUE
var[0].access_rights           READ_WRITE
...
```

Nur synchronisierte Variablen dürfen in Echtzeit-Zyklen geschrieben werden.

Beim Lese-Zugriff einer V.E-Variablen hängt ihr Wert von ihrer Synchronisierung ab:

- Eine synchronisierte Variable wird im Echtzeit-Kontext der CNC ausgewertet.
- Eine nicht-synchronisierte Variable wird zum Zeitpunkt der Dekodierung ausgewertet und als Konstante verwendet.

Programmierbeispiel

Zugriffe auf V.E.-Variablen

```
; Variablen vorbelegen
V.E.SYNC = 47 ; synchrone Variable
V.E.ASYNC = 11 ; asynchrone Variable

; Echtzeit-Zyklus definieren
#RT CYCLE [SCOPE = PROG]

  ; nur einmal durch Anweisungen laufen
  $IF ONCE 1 < 2

    ; neuen Wert zuweisen
    V.E.SYNC = 99

    ; hat den Wert 198
    V.E.VALUE1 = 2 * V.E.SYNC

    ; V.E.ASYNC hat in diesem Echtzeit-Zyklus immer den Wert 11
    ; rechte Seite hat den Wert 22
    V.E.VALUE2 = 2 * V.E.ASYNC

  $ENDIF

#RT CYCLE END

; Hauptprogramm beenden
M30
```

3.2 Echtzeit-Variablen

Variablen mit dem Präfix **V.RT*** sind Echtzeit-Variablen. Sie erlauben Zugriff auf den Echtzeit-Kontext der Steuerung. Sie sind synchronisiert und können sowohl aus dem gewohnten Kontext als auch aus einem Echtzeit-Zyklus gelesen und beschrieben werden. Dabei sind

- **V.RTG.*** kanalspezifische Variablen und
- **V.RTA.*** achsspezifische Variablen.

3.2.1 Kanalspezifische Echtzeit-Variablen

Variablenname	Bedeutung	Datentyp	Einheit	Erlaubter Zugriff	
				Dekoder	Echtzeit-Zyklus
V.RTG.TIMER[]	Timer-Wert für Echtzeit-Kontext, siehe Kapitel Echtzeit-Variablen [▸ 22].	UNS32	ms	L	L
V.RTG.CYCLES.DIAG_LEVEL	Diagnose-Level für Echtzeit-Zyklen, siehe Kapitel Diagnose. [▸ 38]	SGN32	-	L/S	L/S
V.RTG.OVERRIDE.VEL.CYCLE	Geschwindigkeitsoverride aus Echtzeit-Zyklen	UNS16	%	L	L/S
V.RTG.OVERRIDE.VEL.TOTAL	Geschwindigkeitsoverride kombiniert aus allen Einflüssen	UNS32	%	L	L
V.RTG.MEAS_DELTA	Delta zwischen programmiertem und tatsächlichem Kantenstoß	REAL64	mm	L/S	-
V.RTG.LOOP.ENABLED ab V3.1.3105.01	Schleifenbedingung für eine Echtzeit-Schleife	BOOL	-	L/S	L/S
V.RTG.LOOP.COUNT ab V3.1.3105.01	Anzahl der ausgeführten Echtzeit Schleifen	SGN32	-	L/S	L/S

Programmierbeispiel

Bahn-Override reduzieren

```
; Echtzeit-Zyklus definieren
#RT CYCLE [SCOPE = PROG]

  ; ACS-Position der X-Achse abfragen
  ; Override muss zyklisch beschrieben werden, also ohne ONCE
    $IF V.RTA.ACS.ACT_POS.X > 200

      ; Kanal-Override auf 75% reduzieren
      V.RTG.OVERRIDE.VEL.CYCLE = 75

$ENDIF

#RT CYCLE END

; X-Achse auf 500mm fahren
; ab 200mm wird langsamer gefahren
G00 X500

; Hauptprogramm beenden
M30
```

3.2.2 Achsspezifische Echtzeit-Variablen

Variablenname	Bedeutung	Datentyp	Einheit	Erlaubter Zugriff	
				Dekoder	Echtzeit-Zyklus
V.RTA.FIXED_STOP.ACTIVE.X	Fahren auf Festanschlag aktiv	Boolean	-	L	L
V.RTA.FIXED_STOP.DETECTED.X	Fahren auf Festanschlag erkannt	Boolean	-	L	L
V.RTA.FIXED_STOP.ACS.POS.X	Erfasste Anschlagposition im Achskoordinatensystem	SGN64	[mm, inch]	L	L

Variablenname	Bedeutung	Datentyp	Einheit	Erlaubter Zugriff	
				Dekoder	Echtzeit-Zyklus
V.RTA.OVERRIDE.VEL. CYCLE	Geschwindigkeitsoverride aus Echtzeit-Zyklen, für unabhängige Achsbewegung.	UNS16	%	L	L/S
V.RTA.OVERRIDE.VEL. TOTAL	Gesamtgeschwindigkeitsoverride der Achse	UNS16	%	L	L
V.RTA.ACS.ACT_POS	Aktuelle Sollposition der Achse im Achskoordinatensystem	SGN64	mm	L	L
V.RTA.ACS.CUR_POS	Aktuelle Istposition der Achse im Achskoordinatensystem	REAL64	mm	L	L
V.RTA.ACS.CMD_POS	Aktuelle Zielposition der Achse im Achskoordinatensystem	REAL64	mm	L	L
V.RTA.MCS.ACT_POS	Aktuelle Sollposition der Achse im Maschinenkoordinatensysten	SGN32	mm	L	L
V.RTA.MCS.CUR_POS	Aktuelle Istposition der Achse im Maschinenkoordinatensystem	SGN32	mm	L	L
V.RTA.MCS.CMD_POS	Aktuelle Zielposition der Achse im Maschinenkoordinatensystem	SGN32	mm	L	L
V.RTA.MCS.DIST_TO_GO	Aktueller Restweg der Achse im Maschinenkoordinatensystem	SGN32	mm	L	L
V.RTA.PCS.ACT_POS	Aktuelle Sollposition der Achse im Programmierkoordinatensystem	SGN32	mm	L	L
V.RTA.PCS.CUR_POS	Aktuelle Istposition der Achse im Programmierkoordinatensystem	SGN32	mm	L	L
V.RTA.PCS.CMD_POS	Aktuelle Zielposition der Achse im Programmierkoordinatensystem	SGN32	mm	L	L
V.RTA.PCS.DIST_TO_GO	Aktueller Restweg der Achse im Programmierkoordinatensystem	SGN32	mm	L	L
V.RTA.IPO.ACT_POS	Aktuelle Sollposition der Achse im momentanen Interpolatorkoordinatensystem	SGN32	mm	L	L
V.RTA.IPO.CUR_POS	Aktuelle Istposition der Achse im momentanen Interpolatorkoordinatensystem	SGN32	mm	L	L

V.RTA.IPO.CMD_POS	Aktuelle Zielposition der Achse im momentanen Interpolatorkoordinatensystem	SGN32	mm	L	L
V.RTA.IPO.DIST_TO_GO	Restweg der Achse im momentanen Interpolatorkoordinatensystem	SGN32	mm	L	L

3.3 Channel-Variablen

Bei V.CH.-Variablen handelt es sich um eigendefinierte, kanalspezifische Variablen.

Hinweis

Für die Nutzung der V.CH.-Variablen muss über den Parameter P-CHAN-00424 entsprechend Speicher reserviert sein.

3.3.1 Anlegen / Löschen

V.CH-Variablen werden in den Listen der V.E-Variablen angelegt. Sie werden durch das Präfix „vch“ gekennzeichnet.

Einzustellende Werte sind:

- Name
- Typ
- Array Größe (bei nicht Arrays)
- Synchronisation

Programmierbeispiel

Anlegen / Löschen

```
vch[0].name                    Var_name
vch[0].type                    SGN08
vch[0].array_size              0
vch[0].synchronisation         TRUE
```

Ein Löschen der Variablen ist nur durch Entfernen aus der Liste und eine Neu-Interpretierung dieser möglich.

3.3.2 Variablentypen

Die V.CH.-Variablen können alle V.E.-Variablen-Typen annehmen. Dies gilt auch für Strukturdefinitionen. Die Basistypen sind:

BOOLEAN / SGN08 / UNS08 / SGN16 / UNS16 / SGN32 / UNS32 / REAL64 / STRING

Hinweis

Alignment

V.CH.-Struktur-Variablen verfügen über das Alignment von V.E.-Variablen. V.CH.-Strukturen können dadurch V.E.-Strukturen vom gleichen Typ gegenseitig zugewiesen werden!

3.3.3 Grundlegende Eigenschaften / Sichtbarkeit / Scope

Die Variablen sind aus allen Ebenen des NC-Programmes les- und schreibbar. Ein Zugriff auf V.CH.-Variablen ist aus der SPS nicht möglich.

3.4 Beeinflussung der Synchronisierung einer Variablen

Wenn eine asynchrone V.CH.-/V.E.-Variable zwingend für einen Echtzeit-Zyklus verwendet werden muss, dann lässt sich Synchronität für den aktuellen NC-Satz erzwingen.

Dies geschieht durch ein Anfügen eines "s" an den V-Knoten der Variable. Dies ist nur für den aktuellen Satz gültig, für alle anderen Sätze bleibt der Zugriff der Variable asynchron!

```
N10 #RT CYCLE [ID=17 SCOPE = PROG]
N20 $IF ONCE Vs.CH.VarTest1 < 1    ; Hier wird ein synchroner
                                   ; Zugriff erzwungen
N30 Vs.CH.VarTest1 = 1             ; Hier wird ein synchroner
                                   ; Zugriff erzwungen
N40 $ENDIF
N50 #RT CYCLE END
N60 M30
```

Die Synchronisierung ist ebenfalls außerhalb eines Echtzeit-Zyklus möglich.

```
N10 #RT CYCLE [ID=17 SCOPE = GLOBAL]
N20 $IF ONCE 1 < 2
N30 V.E.LEVEL_1_A[0].REAL64 = V.E.LEVEL_1_A[0]
N40 $ENDIF
N50 #RT CYCLE END

N60 Vs.E.LEVEL_1_A[0].REAL64 ; synchroner Zugriff
N70 M30
```

4 Verwendbare Funktionalität

4.1 Timer-Funktionalität

Die **#TIMER**-Funktionalität ist für Echtzeit-Zyklen verfügbar. Die gemessenen Zeiten werden in den Echtzeit-Variablen **V.RTG.TIMER[]** abgelegt. Die Echtzeit-Timer sind verschieden von den Dekoder-Timern.

Programmierbeispiel

#TIMER-Funktionalität

```
; Variablen ausgeben
$FOR P1 = 0, 4, 1
#MSG SAVE EXCLUSIVE ["V.RTG.TIMER[%d] = %f", P1, V.RTG.TIMER[P1]]
$ENDFOR

; Echtzeit-Zyklus definieren
#RT CYCLE [ID = 17 SCOPE = PROG]
  $IF ONCE 1 < 2
    #TIMER START SYN [ID = 0] ; Timer 0 starten
    #TIMER START SYN [ID = 2] ; Timer 2 starten
  $ENDIF
#RT CYCLE END
#FLUSH WAIT

; etwas warten
#TIME 1.5

; Echtzeit-Zyklus definieren
#RT CYCLE [ID = 18 SCOPE = PROG]
  $IF ONCE 1 < 2
    #TIMER STOP SYN [ID = 0] ; Timer 0 stoppen
    #TIMER READ SYN [ID = 0] ; Timer 0 auslesen
    #TIMER READ SYN [ID = 2] ; Timer 2 auslesen, ohne stoppen
  $ENDIF
#RT CYCLE END
#FLUSH WAIT

; Variablen ausgeben
$FOR P1 = 0, 4, 1
  #MSG SAVE EXCLUSIVE ["V.RTG.TIMER[%d] = %f", P1, V.RTG.TIMER[P1]]
$ENDFOR

; Hauptprogramm beenden
M30
```

Achtung

Es sind nur synchrone **#TIMER**-Befehle erlaubt. Alle **#TIMER**-Befehle in Echtzeit-Zyklen müssen mit dem Schlüsselwort **SYN** als synchron markiert werden.

4.2 Ausgabe von Benutzerfehlern

Der NC-Befehl #ERROR ermöglicht benutzerdefinierte Fehlermeldungen innerhalb eines Echtzeit-Zyklus. Die Syntax ist in [PROG//Benutzerdefinierte Fehlerausgabe (#ERROR)] beschrieben.

Hinweis

Bei Ausgabe eines Fehlers mit RC >= 1 geht der NC-Kanal in den Fehlerzustand und mögliche Bahnbewegungen werden gestoppt. Bei Ausgabe von Warnungen (RC = 0) werden die Bahnbewegungen fortgesetzt.

Programmierbeispiel

#ERROR- Befehl innerhalb eines Echtzeitzyklus

```
; RT-Zyklus definieren; Fehler ausgeben sobald X > 99
#RT CYCLE [SCOPE = PROG]
  $IF ONCE V.RTA.ACS.ACT_POS.X > 99
    #ERROR [ID=666 RC=2 PM1=1 PV1=99]
  $ENDIF
#RT CYCLE END
```

4.3 M- und H-Technologie-Funktionen

Aus dem Echtzeit-Zyklus heraus können Technologie-Funktionen ausgegeben werden. Dabei gelten die folgenden Regeln:

- Es können nur kanalspezifische Technologiefunktionen ausgegeben werden, achsspezifische jedoch nicht.
- Es können nur Technologiefunktionen ausgegeben werden, die in den Kanalparametern definiert sind. Alle anderen führen zu einem Fehler.
- Alle Technologiefunktionen werden als MOS (ohne Synchronisierung) ausgegeben, unabhängig von der konfigurierten Synchronisationsart.

Die Echtzeit-Technologiefunktionen werden in einem neu angelegten Bereich auf dem HLI ausgegeben, **nicht** im Bereich der klassischen Technologiefunktionen. Sie sind auf dem HLI mit der neuen Kennung **POS_RT** versehen.

Die Ausgabe von Echtzeit-Technologiefunktionen auf dem HLI geschieht im gleichen CNC-Takt, in dem sie beauftragt werden. Auf freie Ressourcen wird nicht gewartet. Wenn Echtzeit-Technologiefunktionen nicht takttreu ausgegeben werden können, weil z.B. das HLI belegt ist, führt dies zu einem Fehler. Die Echtzeit-Technologiefunktionen werden in der Reihenfolge auf das HLI gelegt, in der sie von den Echtzeit-Zyklen beauftragt werden. Sie werden innerhalb eines Taktes durchnummeriert und mit einem Zeitstempel versehen.

Für ein Funktionieren hat der Anwender für folgende Punkte Sorge zu tragen:

- Die SPS muss einen Task im CNC-Takt haben, der die Echtzeit-Technologiefunktionen ausliest.
- Die SPS muss in jedem Takt genügend Echtzeit-Technologiefunktionen quittieren, um genug HLI-Plätze für Echtzeit-Technologiefunktionen des nächsten CNC-Taktes zu schaffen.

Programmierbeispiel

Ausgabe von Technologie-Funktionen

```
; X-Achse auf 0mm fahren
G00 X0

; Echtzeit-Zyklus definieren
#RT CYCLE [SCOPE = PROG]

  ; ACS-Position der X-Achse abfragen
  $IF ONCE V.RTA.ACS.ACT_POS.X > 200

    ; M-Funktion ausgeben
    ; M100 muss im Kanal konfiguriert sein
    ; wird noch in diesem Takt ausgegeben
    ; wird ohne Synchronisation (MOS) ausgegeben
    M100

  $ENDIF

#RT CYCLE END

; X-Achse auf 300mm fahren
G00 X300

; Hauptprogramm beenden
M30
```

4.4 Spindelprogrammierung

In Echtzeit-Zyklen ist eine Spindelprogrammierung möglich. Die Programmierung von Haupt- als auch Nebenspindel werden unterstützt.

Hinweis

Belegte Ressourcen

Die Spindelaufträge aus Echtzeit-Zyklen werden über Standardmechanismen der CNC ausgegeben. Im Unterschied zur Spindelprogrammierung aus dem Dekoder-Kontext (Standardverhalten) wird aber nicht gewartet, bis genügend Ressourcen für die Beauftragung vorhanden sind. Steht für die Echtzeit-Beauftragung kein Pufferplatz zur Verfügung oder ist das Interface zur Spindel noch durch vorherige Aufträge belegt, dann führt dies zu einem Fehler.

Die Programmierung muss vollständig in einem Satz erfolgen. So ist es nicht erlaubt, erst die Drehrichtung mit **M3** festzulegen und in einem späteren Satz die Drehzahl zu programmieren. **M3** und Drehzahl müssen im selben Satz programmiert sein.

4.4.1 Spindelrehzahl- und Richtung für Endlosdrehen

Das Endlosdrehen einer Spindel kann über gleichzeitige Programmierung von **M3**/**M4** und

- dem **S**-Wort für Hauptspindeln
- dem **REV**-Parameter für Nebenspindeln

erfolgen.

Programmierbeispiel

Endlos-Drehen

```
; Echtzeit-Zyklus definieren
#RT CYCLE [...]

  ; externe Trigger-Bedingung
  $IF ONCE V.E.TRIGGER == 1

    ; Hauptspindel beauftragen
    M03 S1000

    ; Nebenspindel beauftragen
    S2[M03 REV=1000]

  $ENDIF

#RT CYCLE END

; ...

; Hauptprogramm beenden
M30
```

4.4.2 Spindeln positionieren

Für die Spindelpositionierung müssen

- **M3**/**M4** für die Bewegungsrichtung,
- **M19** für die Spindelpositionierung,
- **S**/**REV** für die Drehgeschwindigkeit und
- **S.POS**/**POS** für die Zielposition

programmiert werden.

Programmierbeispiel

Spindel positionieren

```
; Echtzeit-Zyklus definieren
#RT CYCLE [...]

  ; externe Trigger-Bedingung
  $IF ONCE V.E.TRIGGER == 1

    ; Hauptspindel beauftragen
    M03 M19 S1000 S.POS314

    ; Nebenspindel beauftragen
    S2[M03 M19 REV=1000 POS=314]

  $ENDIF

#RT CYCLE END

; ...

; Hauptprogramm beenden
M30
```

4.4.3 Spindelstopp

Der Spindelstopp wird mit **M5** programmiert.

Programmierbeispiel

Spindelstopp

```
; Echtzeit-Zyklus definieren
#RT CYCLE [...]

  ; externe Trigger-Bedingung
  $IF ONCE V.E.TRIGGER == 1

    ; Hauptspindel beauftragen
    M05

    ; Nebenspindel beauftragen
    S2[M05]

  $ENDIF

#RT CYCLE END

; ...

; Hauptprogramm beenden
M30
```

4.4.4 PLC-Spindeln

Eine PLC-Spindel wird durch den Kanal-Parameter P-CHAN-00069 **plc_control** gekennzeichnet.

Programmierbeispiel

PLC-Spindel konfigurieren

```
...
spindel[2].plc_control 1
spindel[2].bezeichnung S3
spindel[2].log_achs_nr 10
...
```

Die Programmierung einer PLC-Spindel im Echtzeit-Zyklus unterscheidet sich nicht von der Programmierung einer CNC-Spindel. Die Beauftragungen werden direkt an die PLC weitergegeben. Alle beteiligten Techno-Funktionen werden mit dem Synchronisationstyp MOS ausgegeben.

4.5 Einzelachsbewegungen

Für die Verwendung von Einzelachsbewegung in Echtzeit-Zyklen muss die Schnittstelle der Kanalachse über P-AXIS-00457 freigeschaltet sein.

4.5.1 Programmiersyntax

Syntax:

<Achsname> **[INDP ABORTING** | **BUFFERED** [**OFFSET**=..] **G90 | G91 G00 | G01** [**FEED**=..] [**POS**=..] [**DIR**=..] | **STOP]**

<Achsname>	Name der unabhängigen Achse
INDP	Kennung für eine unabhängige Achse
ABORTING / BUFFERED	ABORTING bricht eine zuvor gestartete Achsbewegung der programmierten Achse ab. Wird das Schlüsselwort nicht angegeben, dann wird standardmäßig ABORTING verwendet. **Hinweis:** BUFFERED noch nicht verfügbar.
STOP	Stopp der Achse, Abbruch des aktuellen Bewegungsauftrags. Nicht mit anderen Schlüsselworten kombinierbar
OFFSET	Angabe, welche Achsversätze mit eingerechnet werden sollen; s. Offsettabelle
G90 / G91	Absolut- / Relativmaß
G00 / G01	Eilgang- / Linearinterpolation
FEED	Achsspezifischer Vorschub in [mm/min, m/min, inch/min] Wird FEED ohne die Angabe von Richtung und einer G-Funktion G00/G01 programmiert, löst es eine Endlosbewegung der Achse mit dem vorgegebenen Vorschub aus.
POS	Achsposition in [mm, inch]
DIR	Richtungsangabe, zulässige Angaben: • POS, positive Richtung • CUR, aktuelle Richtung • NEG, negative Richtung Es können auch mathematische Ausdrücke verwendet werden, die zu einem der Werte resultieren.

Hinweis

Bei der Verwendung von ‚POS' muss zwingend ‚G90' oder ‚G91' angegeben werden.

Fehlt diese Angabe, so wird der Fehler ID 50967 ausgegeben.

Schlüsselwort für Offset	Bedeutung
ALL	Alle aktiven Versatzmaße der Achse
ZERO	Nullpunktverschiebungen
ADD_ZERO	Additive Nullpunktverschiebungen bzw. Bezugspunkt-verschiebungen
PSET	Istwertverschiebungen
CLAMP	Platzversätze
TOOL	Werkzeugversätze
MEASURE	Messverschiebungen
MANUAL	Handbetriebverschiebungen

Programmierbeispiel

Positionierung mit Abbruchbedingung

```
; Aufgabenstellung:
; Sobald die X-Achse über Position 100 fährt, wird die Z-Achse
; auf Position 900 bewegt.
; Sollte die Z-Achse sich bereits bewegen, wird diese Bewegung
; abgebrochen
; Die Bewegung wird mit Einberechnung aller Offsets durchgeführt.
N010 #RT CYCLE [ID=2 SCOPE=PROG]
N020   $IF ONCE V.RTA.ACS.ACT_POS.X > 100
N030     Z [INDP ABORTING G01 G90 FEED=500 POS=900 OFFSET=ALL]
N040   $ENDIF
N050 #RT CYCLE END
N060 G90 X100
N070 G90 X200
N080 M30
```

Programmierbeispiel

Endlosdrehen mit Startbedingung

```
; Aufgabenstellung:
; Z positiv endlos drehen, wenn X > 1mm

; Echtzeit-Zyklus definieren
N10 #RT CYCLE [SCOPE = PROG]
N20   $IF ONCE V.RTA.ACS.ACT_POS.X > 1
N30     Z[INDP DIR = POS FEED = 1000]
N40   $ENDIF
N50 #RT CYCLE END

; Bewegung starten
N60 G01 X100

; beenden--
N70 M30
```

4.5.2 Positionsanforderung

Nach Löschen eines Echtzeit-Zyklus mit dem **#RT CYCLE DELETE** erfolgt eine Positionsanforderung.

Bei Echtzeit-Zyklen mit der Gültigkeit SCOPE = BLOCK erfolgt nach Verlassen der Gültigkeit eine Positionsanforderung, wenn im NC-Programm eine Achse programmiert wird, für die zuvor im Echtzeit-Zyklus eine unabhängige Achsbewegung programmiert wurde.

Programmierbeispiel

Positionsanforderung Echtzeit-Zyklus

```
N010 G00 X0 Y0 Z0 F500 G70
N020 #FLUSH WAIT

; Echtzeit-Zyklus aktivieren und Z verfahren
N030 #RT CYCLE [SCOPE = BLOCK END_ACTION = MOVE_ABORT]
N040   $IF ONCE V.RTA.ACS.ACT_POS.X > 10
N050     Z[INDP ABORTING G0 G90 POS = 137]
N060   $ENDIF
N070 #RT CYCLE END

; Dies ist der Block, in dem der Echtzeit-Zyklus ausgefuehrt wird.
N080 G01 X100 F1000
N090 G01 X-100 F1000
N100 G01 X100 F1000

; Hier findet eine Positionsanforderung statt,
; da Z bewegt wurde
N120 G01 Z100 F1000
; Hier ist der Echtzeit-Zyklus bereits außerhalb seiner Gültigkeit
; Er wird also beendet und alle Achsbewegungen, die durch ihn
; verursacht wurden, werden abgebrochen.

N130 G01 Z-100 F1000
N140 G01 Z100 F1000

N150 M30
```

4.5.3 Maßeinheit

Die Maßeinheit in einem Echtzeit-Zyklus wird zum Zeitpunkt seiner Definition festgelegt. Es handelt sich um die zum Zeitpunkt der Definition gültige Maßeinheit.

Bei einem Wechsel der Maßeinheit innerhalb des NC-Programms, in dem Echtzeit-Zyklen zum Einsatz kommen, erfolgt die Umschaltung beim Start des Echtzeit-Zyklus.

Programmierbeispiel

Einheiten umschalten bei Echtzeit-Zyklen

```
N010 G00 X0 Y0 Z0 F500 G71
N020 #FLUSH WAIT

; Echtzeit-Zyklus aktivieren und Z verfahren,
; hier wird die Position in mm verfahren
N030 #RT CYCLE DEF[ID = 15 SCOPE = BLOCK END_ACTION = MOVE_ABORT]
N040   $IF ONCE V.RTA.ACS.ACT_POS.X > 10
N050     Z[INDP ABORTING G0 G90 POS = 137 ]
N060   $ENDIF
N070 #RT CYCLE END

; hier wird auf inch umgeschaltet
N080 G70
; hier wird der Echtzeit-Zyklus gestartet, aber Positionen
; im Echtzeit-Zyklus sind in mm
N090 #RT CYCLE START [ID = 15]

; Dies ist der Block, in dem der Echtzeit-Zyklus ausgefuehrt wird.
N100 G01 X100 F1000
N110 G01 X-100 F1000
N120 G01 X100 F1000

; Hier findet eine Positionsanforderung statt,
; da Z bewegt wurde
N130 G01 Z100 F1000
N140 G01 Z-100 F1000
N150 G01 Z100 F1000

N160 M30
```

5 Diagnose

Wenn die Funktionalität Echtzeit-Zyklen aktiviert ist, dann werden beim Erstellen der Diagnose-Daten Informationen der Echtzeit-Zyklen mit ausgegeben.

Beispiel

Beispiel von Diagnosedaten

```
BAHN : ECHTZEIT-ZYKLEN DIAGNOSE DATEN KANAL-NR.: 1
==================================================================

Hinweis:
Bei der Diagnose werden evtl. einige Nachrichten ausgeblendet.
Bitte konsultieren Sie die Dokumentation der Echtzeit-Zyklen, um zu er-
fahren, wie die Diagnose beeinflusst werden kann.

Zeitstempel           Level              Nachricht
------------------------------------------------------------------
  50024    INFO    Echtzeit-Zyklen-Manager wurde initialisiert
  50024    INFO    Echtzeit-Zyklen-Manager: 0 Zyklen
  426098   INFO    Zyklus 1 mit Gueltigkeit PROG wurde angelegt
  426098   INFO    Zyklus 1 wurde gestartet
...
```

Jede Zeile ist mit einem Diagnose-Level versehen. Der Anwender kann über die Echtzeit-Variable **V.RTG.CYCLES.DIAG_LEVEL** die Ausgabe beeinflussen. Je höher der Wert der Variable ist, umso mehr Informationen werden ausgegeben. Folgende Diagnose-Level sind einstellbar.

Wert	Beschreibung
0	Keine Diagnosedaten.
1	Fehler aus Echtzeit-Zyklen werden ausgegeben.
2	Warnungen, die auf ein Problem hindeuten können.
3	Informationen zur Verwaltung von Echtzeit-Zyklen, Default.
4	Informationen zu Aktionen der Echtzeit-Zyklen.

Mit dem voreingestellten Wert **V.RTG.CYCLES.DIAG_LEVEL = 3**, werden alle Informationen der Stufen 0 bis 3 ausgegeben.

Programmierbeispiel

Diagnose-Level einstellen

```
; Diagnose sehr hoch einstellen
; alle Informationen werden ausgegeben
V.RTG.CYCLES.DIAG_LEVEL = 99

; Echtzeit-Zyklus definieren
#RT CYCLE [SCOPE = PROG]

  ; Aktionen
  ; ...

#RT CYCLE END

; ...

; Hauptprogramm beenden
M30
```

6 Parameter

6.1 Übersicht Kanal- und Achsparameter

ID	Beschreibung
P-CHAN-00406	Echtzeit-Zyklen-Funktionalität einschalten
P-CHAN-00407	Speicher für Echtzeit-Zyklen
P-CHAN-00424	Speichergröße für V.CH.-Variablen
P-CHAN-00425	Max. Ausführungsdauer der Echtzeit-Zyklen pro CNC-Takt
P-CHAN-00426	Anzahl der Elementar-Anweisungen für Zeitprüfung
P-CHAN-00427	Max. Anzahl der Elementar-Anweisungen pro CNC-Takt
P-CHAN-00480	Max- Anzahl von Aktionen innerhalb eines Echtzeit-Zyklus

ID	Beschreibung
P-AXIS-00457	Freischalten der PLCopen-Schnittstelle einer Kanalachse

6.2 Kanalparameter

P-CHAN-00406	Aktivierung Echtzeit-Zyklen
Beschreibung	Mit diesem Parameter kann die Funktionalität der Echtzeit-Zyklen im NC-Kanal aktiviert werden. Für die Übernahme der Änderung ist ein Neustart der Steuerung notwendig. Beispiel: `configuration.rt_cycles.enable 1`
Parameter	configuration.rt_cycles.enable
Datentyp	BOOLEAN
Datenbereich	0/1
Dimension	----
Standardwert	0
Anmerkungen	Parameter ist ab V3.1.3107.10 verfügbar. Die Verwendung des Parameters „rt_cycles.enable“ `rt_cycles.enable 1` (ab V3.1.3105) wird weiterhin unterstützt.

P-CHAN-00407	Speichergröße für Echtzeit-Zyklen
Beschreibung	Mit diesem Parameter kann für die Echtzeit-Zyklen der Speichergröße festgelegt werden. Die Angabe der Speichergröße erfolgt in Byte. Für die Übernahme der Änderung ist ein Neustart der Steuerung notwendig. Anschließend steht für die Echtzeit-Zyklen der angegebene Speicher zusätzlich zur Verfügung. Beispiel: `configuration.rt_cycles.memory 60000`
Parameter	configuration.rt_cycles.memory
Datentyp	UNS32
Datenbereich	0 ... MAX(UNS32) - 1
Dimension	----
Standardwert	48000
Anmerkungen	**Hinweis:** Die Belegung von P-CHAN-00407 ist nur erforderlich, wenn der standardmäßig eingestellte Speicher durch Aktivierung der Echtzeit-Zyklen (P-CHAN-00406 [▸ 41]) nicht mehr ausreicht. Parameter ist ab V3.1.3107.10 verfügbar. Die Verwendung des Parameters „rt_cycles.memory“ `rt_cycles.memory 60000` (ab V3.1.3105) wird weiterhin unterstützt.

P-CHAN-00424	**Speichergröße für V.CH.-Variablen**
Beschreibung	Dieser Parameter legt die Speichergröße in Byte für die V.CH.-Variablen fest. Beispiel: `configuration.decoder.v_ch_memory 10000`
Parameter	configuration.decoder.v_ch_memory
Datentyp	UNS32
Datenbereich	0 ... MAX(UNS32) - 1
Dimension	Byte
Standardwert	0
Anmerkungen	Der Speicher beinhaltet Nutzdaten sowie interne Verwaltungsdaten. Dies führt dazu, dass der effektiv verfügbare Nutzspeicher immer kleiner ist als der eingestellte Wert. Parameter ab Version V3.1.3107.10 verfügbar. Die Verwendung des Parameters „v_ch_memory“ direkt ohne Struktur (ab V3.1.3104) wird weiterhin unterstützt, sollte aber in neuen Applikationen nicht mehr verwendet werden. `v_ch_memory 10000`

P-CHAN-00425	**Max. Ausführungsdauer der Echtzeit-Zyklen pro CNC-Takt**
Beschreibung	Mit diesem Parameter kann die maximale Ausführungsdauer der Echtzeit-Zyklen im NC-Kanal festgelegt werden. Die Angabe erfolgt in Prozent (%) und bezieht sich auf die Dauer eines CNC-Taktes. Beispiel: Wenn die Echtzeit-Task der CNC mit 2ms getaktet ist und der Parameter P-CHAN-00425 auf 75 steht, dann dürfen die Echtzeit-Zyklen insgesamt maximal 1.5ms Ausführungszeit benötigen. Wird diese Zeit überschritten, dann wird der Fehler ID 50939 ausgegeben.
Parameter	rt_cycles.max_duration
Datentyp	UNS16
Datenbereich	0 < P-CHAN-00425 < MAX_UNS16
Dimension	%
Standardwert	75
Anmerkungen	Der Anwender ist bezüglich der Anzahl der Anweisungen innerhalb eines Echtzeit-Zyklus nicht beschränkt. Wenn Echtzeit-Zyklen zu viele Anweisungen enthalten und nicht in einem CNC-Takt ausgeführt werden können, kann es zu Echtzeit-Überschreitungen kommen. Dieser Parameter stellt zusammen mit P-CHAN-00426 und P-CHAN-00427 einen Sicherheitsmechanismus dar, um diese Echtzeit-Überläufe möglichst früh zu vermeiden.

P-CHAN-00426	**Anzahl der Elementar-Anweisungen für Zeitprüfung**
Beschreibung	Mit diesem Parameter kann die Anzahl der Elementar-Anweisungen festgelegt werden, nach denen eine erneute Zeitprüfung durchgeführt wird. Für die Ausführungsdauer der Echtzeit-Zyklen muss innerhalb eines CNC-Taktes regelmäßig kontrolliert werden, ob die erlaubte Ausführungszeit bereits überschritten ist. Dafür wird nach einer gegebenen Anzahl von Elementar-Anweisungen eines Zyklus die bereits verbrauchte Zeit geprüft. Der Parameter P-CHAN-00426 gibt die Anzahl dieser Elementar-Anweisungen an.
Parameter	rt_cycles.cont_steps
Datentyp	UNS32
Datenbereich	0 < P-CHAN-00426 < MAX_UNS32
Dimension	----
Standardwert	100
Anmerkungen	Der Anwender ist bezüglich der Anzahl der Anweisungen innerhalb eines Echtzeit-Zyklus nicht beschränkt. Wenn Echtzeit-Zyklen zu viele Anweisungen enthalten und nicht in einem CNC-Takt ausgeführt werden können, kann es zu Echtzeit-Überschreitungen kommen. Dieser Parameter stellt zusammen mit P-CHAN-00425 und P-CHAN-00427 einen Sicherheitsmechanismus dar, um diese Echtzeit-Überläufe möglichst früh zu vermeiden.

P-CHAN-00427	**Max. Anzahl der Elementar-Anweisungen pro CNC-Takt**
Beschreibung	Mit diesem Parameter kann die maximale Anzahl der Elementar-Anweisungen pro CNC-Takt festgelegt werde. Mit P-CHAN-00427 kann zusätzlich zu P-CHAN-00425 und P-CHAN-00426 die Ausführungsdauer in Echtzeit-Zyklen im CNC-Takt beschränkt werden. Überschreitet die Anzahl der Elementar-Anweisungen im aktuellen CNC-Takt den Wert dieses Parameters, dann wird ein Fehler ID 50854 ausgegeben.
Parameter	rt_cycles.max_steps
Datentyp	UNS32
Datenbereich	0 < P-CHAN-00427 < MAX_UNS32
Dimension	----
Standardwert	MAX_UNS32 - 1
Anmerkungen	Der Anwender ist bezüglich der Anzahl der Anweisungen innerhalb eines Echtzeit-Zyklus nicht beschränkt. Wenn Echtzeit-Zyklen zu viele Anweisungen enthalten und nicht in einem CNC-Takt ausgeführt werden können, kann es zu Echtzeit-Überschreitungen kommen. Dieser Parameter stellt zusammen mit P-CHAN-00425 und P-CHAN-00426 einen Sicherheitsmechanismus dar, um diese Echtzeit-Überläufe möglichst früh zu vermeiden.

P-CHAN-00480	Maximale Anzahl von Aktionen im Echtzeit-Zyklus
Beschreibung	Mit diesem Parameter kann die maximale Anzahl möglicher Aktionen innerhalb eines Echtzeit-Zyklus festgelegt werden. Mögliche Aktionen sind Einzelachsbewegung, Spindelbeauftragung, usw. Werden zu viele Aktionen innerhalb eines Echtzeit-Zyklus beauftragt, wird der Fehler ID 51028 ausgegeben.
Parameter	configuration.rt_cycles.buffers
Datentyp	UNS16
Datenbereich	0 ... MAX(UNS16) - 1
Dimension	----
Standardwert	5
Anmerkungen	Parameter ist verfügbar ab V3.1.3107.10

6.3 Achsparameter

P-AXIS-00457	Freischalten der PLCopen-Schnittstelle einer Kanalachse	
Beschreibung	Dieser Parameter schaltet die PLCopen-Schnittstelle einer Kanalachse frei. Danach kann die Achse durch die SPS oder aus dem NC-Programm (s. PROG//PLCopen-Programmierung) mit Verfahrbefehlen (absolute / relative Positionierung, Verfahrbewegungen mit Geschwindigkeit, sowie Anhalten) beauftragt werden. Folgende Bausteine werden unterstützt: • MC_MoveAbsolute • MC_MoveRelative • MC_MoveVelocity • MC_Halt	
Parameter	kenngr.enable_single_axis	
Datentyp	BOOLEAN	
Datenbereich	0/1	
Achstypen	T, R	
Dimension	T: ----	R: ----
Standardwert	0	
Antriebstypen	----	
Anmerkungen	Die Beauftragung ist für Linearachsen und auch für rotatorische Achsen (Modulo-Achsen) möglich, siehe P-AXIS-00015.	

7 Anhang

7.1 Anregungen, Korrekturen und neueste Dokumentation

Sie finden Fehler, haben Anregungen oder konstruktive Kritik? Gerne können Sie uns unter documentation@isg-stuttgart.de kontaktieren. Die aktuellste Dokumentation finden Sie in unserer Onlinehilfe (DE/EN):

QR-Code Link: https://www.isg-stuttgart.de/documentation-kernel/

Der o.g. Link ist eine Weiterleitung zu:

https://www.isg-stuttgart.de/fileadmin/kernel/kernel-html/index.html

Hinweis

Mögliche Änderung von Favoritenlinks im Browser:

Technische Änderungen der Webseitenstruktur betreffend der Ordnerpfade oder ein Wechsel des HTML-Frameworks und damit der Linkstruktur können nie ausgeschlossen werden.

Wir empfehlen, den o.g. „QR-Code Link“ als primären Favoritenlink zu speichern.

PDFs zum Download:

DE:
https://www.isg-stuttgart.de/produkte/softwareprodukte/isg-kernel/dokumente-und-downloads

EN:
https://www.isg-stuttgart.de/en/products/softwareproducts/isg-kernel/documents-and-downloads

E-Mail: documentation@isg-stuttgart.de

Stichwortverzeichnis

P

www.isg-stuttgart.de
support@isg-stuttgart.de