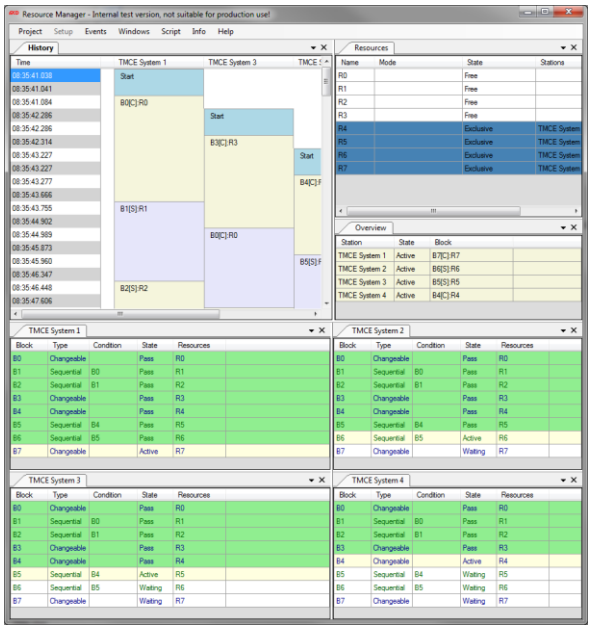
Bedienungsanleitung

Toolmonitor ResourceManager



GET IN touch WITH SENSITIVE TESTING

Softline

Modline

Conline

Boardline

Avidline

Pixline

Applikation

MCD Elektronik GmbH
Hoheneichstr. 52
75217 Birkenfeld
Telefon 0 72 31/78 405-0
Telefax 0 72 31/78 405-10
info@mcd-elektronik.de
www.mcd-elektronik.de
Sitz: Birkenfeld
Geschäftsführer: Bruno Hörter
Registergericht Mannheim
HRB 505692

Vorlagenversion: 5.4 / 2018-04-09
V1.2 2018-04-24 (TS)

Inhalt

1. Allgemein

Der ResourceManager dient im Zusammenspiel mit dem TestManager der Ablaufsteuerung für Testsysteme, bei welchen mehrere Geräte gleichzeitig und asynchron geprüft werden und dabei die vorhandene Hardware für die jeweiligen Messungen gemeinsam verwendet wird.

Folgende Möglichkeiten bestehen durch den Einsatz des ResourceManagers:

- Verriegelung gemeinsam genutzter Hardware
- Optimierung des Testablaufes indem Testblöcke entsprechend der zur Verfügung stehenden Hardware durchgeführt werden
- Gleichzeitiger und asynchroner Test mehrerer Prüflinge
- Gleichzeitiger Test unterschiedlicher Prüflinge (Typen)
- Synchronisation von Testabläufen

Um dies zu ermöglichen wird der Test in einzelne Testblöcke unterteilt. Für diese Testblöcke können Regeln erstellt werden, wie diese Testblöcke abgearbeitet und getauscht werden können.

Bestellnummer: # 150039

2. Konfiguration der Testblöcke

Die Konfiguration der Testblöcke erfolgt im Testeditor des TestManagers. Über einen neuen Filter können im Testeditor Testblöcke angelegt werden.

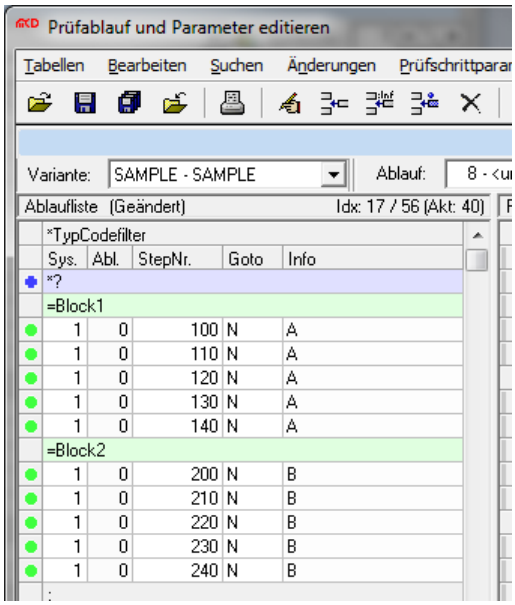


Abbildung 1: Testblöcke in der Testsequenz

Ein Testblock bzw. Blockfilter beginnt mit dem Zeichen „=“ gefolgt von einem für den TestManager beliebigen Text. Dieser „beliebige Text“ muss für das Zusammenspiel mit dem ResourceManager einer definierten Syntax entsprechen, welche weiter unten beschrieben wird.

Ein Blockfilter gilt für alle nachfolgenden Testschritte der Testsequenz bis ein neuer oder leerer Blockfilter definiert wird. Ein Blockfilter kann im Testablauf mehrfach verwendet werden. Im Zusammenspiel mit dem ResourceManager ist dies zwar möglich aber nicht sehr übersichtlich und daher nicht zu empfehlen.

3. Neue Systemschritte im TestManager

Zur Aktivierung der Blockfilter und für das Zusammenspiel mit dem ResourceManager wurden im TestManager drei neue Systemschritte eingeführt:

- Block - Start
- Block - End
- Block - Change

3.1. Block - Start

Nach jedem Prüflingsstart aber noch vor der eigentlichen Testsequenz wird der Block - Start - Step aufgerufen.

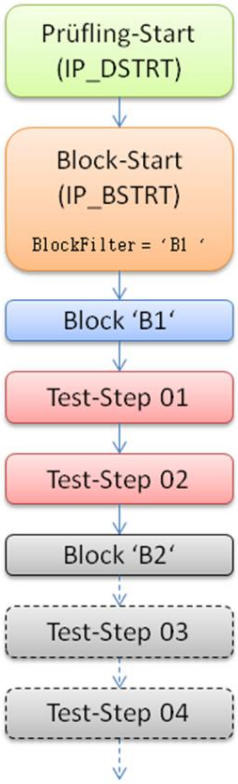


Abbildung 2: Block - Start - Step

In diesem Block - Start - Step kann ein Blockfilter aktiviert werden. Im nachfolgenden Testablauf werden dann die Schritte ausgeführt, zu welchen der Blockfilter passt. Testschritte, welche keinem Blockfilter zugeordnet sind, werden immer ausgeführt. Wird im Block - Start kein Blockfilter festgelegt, so werden alle Testschritte ausgeführt.

In der Grafik oben wird im Block - Start der Blockfilter auf „B1“ festgelegt und somit werden in der nachfolgenden Abarbeitung auch nur die Schritte „Test - Step 01“ und „Test - Step 02“ ausgeführt.

3.1.1. SetSeqBlock

Um den Block festzulegen wurde ein neuer Befehl im TestManager eingeführt:

```
procedure SetSeqBlock(sBlockName : string);
sBlockName : Name des aktiven Blockfilters
```

3.1.2. GetSeqBlock

Zur Abfrage des aktuellen Blockfilters dient der Befehl:

```
function GetSeqBlock : string;
```

3.1.3. ResetSeqBlock

Zurückgesetzt wird der Blockfilter mit dem Befehl:

```
procedure ResetSeqBlock;
```

3.1.4. GetSeqBlocks

Folgender Befehl dient der Abfrage aller definierten Blöcke:

```
function GetSeqBlocks : stringvector;
```

Diese Funktion gibt eine Liste aller noch definierten Blöcke zurück.

3.1.5. Beispiel für Block - Start - Step

Im einfachsten Fall, hat der Block - Start - Step folgenden Aufbau:

```
step

    SetSeqBlock('B1');

end.
```

3.2. Block - End

Ist das Ende der Testsequenz erreicht, so wird vor dem Prüflingsende der Block-End-Step aufgerufen. In diesem Block-End-Step kann festgelegt werden, ob der Testablauf fertig abgearbeitet wurde oder die Testsequenz nochmals durchlaufen werden soll.

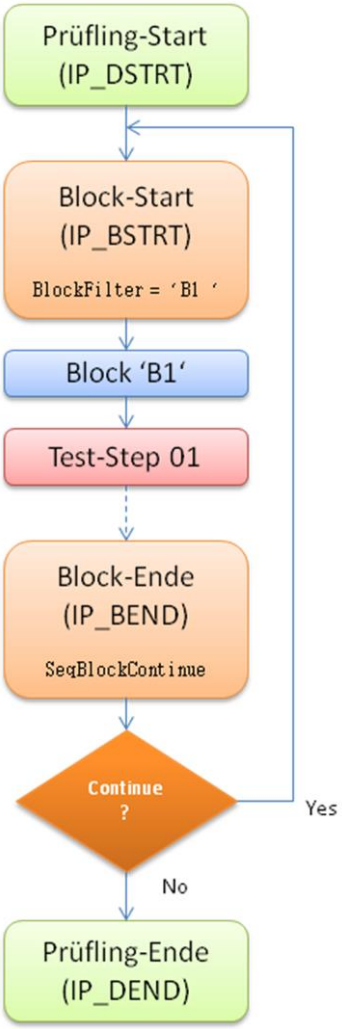


Abbildung 3: Block - End - Step

3.2.1. SeqBlockContinue

Um festzulegen, ob ein weiterer Block abgearbeitet werden soll, wurde ein neuer Befehl im TestManager eingeführt:

```
procedure SeqBlockContinue;
```

Wird dieser Befehl aufgerufen, so werden im Anschluss an den Block - End - Step wieder der Block - Start - Step und die Testsequenz abgearbeitet.

3.2.2. SeqBlockCanContinue

In Ausnahmefällen, kann die Testsequenz nicht noch einmal ausgeführt werden. Dies ist zum Beispiel bei einem Abbruch der aktuellen Prüfung der Fall. Wird hier der Befehl „SeqBlockContinue" aufgerufen, so hat dies keine Auswirkungen und der Test wird nicht wiederholt.

Dazu kann folgender Befehl verwendet werden:

```
function SeqBlockCanContinue : real;
```

Dieser Befehl gibt den Wert „1" oder „true" zurück, wenn ein weiterer Block geprüft werden kann.

3.2.3. GetRemainingSeqBlocks

Um abzufragen, welche Blöcke noch nicht abgearbeitet wurden kann folgender Befehl verwendet werden:

```
function GetRemainingSeqBlocks : stringvector;
```

Diese Funktion gibt eine Liste aller noch nicht durchgeführten Blöcke zurück.

3.2.4. Beispiel für Block - End - Step

Im einfachsten Fall, hat der Block - End - Step folgenden Aufbau:

```
Step
  if SeqBlockCanContinue and (Len(GetRemainingSeqBlocks) > 0) then begin
    SeqBlockContinue;
  end;
end.
```

3.3. Block - Change

Im Zusammenspiel mit dem ResourceManager ist es wichtig darauf zu reagieren, wenn sich der aktuelle Block ändert. Im normalen Testablauf geschieht dies nur im Block - Start - Step, im Step - By - Step Modus kann dies aber jederzeit geschehen.

Um hier reagieren zu können, wurde der Block - Change - Step eingeführt. Dieser wird immer dann aufgerufen, wenn sich der aktuelle Block ändert.

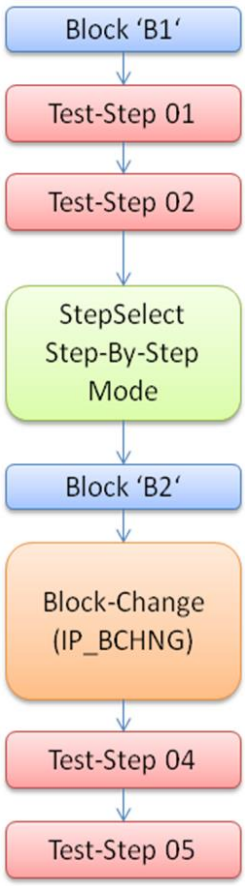


Abbildung 4: Block - Change - Step

Im Block - Change - Step kann z.B. der ResourceManager über den erzwungen Wechsel des aktuellen Blocks informiert werden oder es wird im Block - Change - Step darauf gewartet, dass der ResourceManager den betreffenden Block zur Verfügung stellt bzw. freigibt, wenn alle notwendigen Ressourcen zur Verfügung stehen.

4. Einführung in die Bedienung des ResourceManagers

4.1. Kurzeinführung

Wie in der Einleitung beschrieben, dient der ResourceManager der Verriegelung gemeinsam genutzter Hardware und der Freigabe von Ablaufblöcken.

4.2. Setup

Es ist keinerlei Konfiguration des ResourceManagers erforderlich, der ResourceManager „lernt“ zu Laufzeit alle Instanzen, definierten Blöcke und Ressourcen selbständig ein.

Lediglich der verwendete Anzeige - Font und die Größe der History können bei Bedarf den eigenen Anforderungen entsprechend angepasst werden.

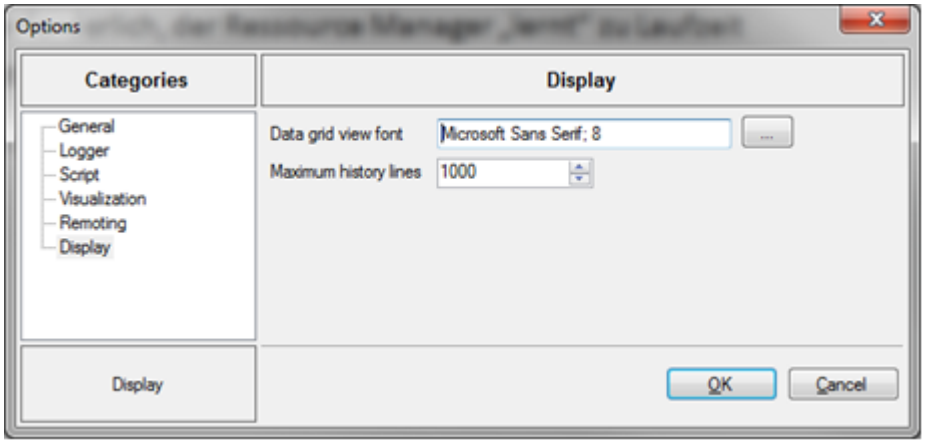


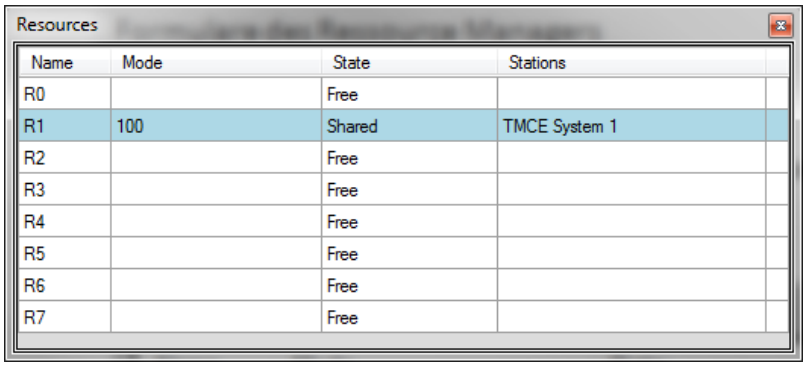
Abbildung 5: Setup des ResourceManagers

Wie bei allen Toolmonitoren wird das eingestellte Layout (Formularpositionen und Spaltenbreiten) ebenfalls im Setup gespeichert.

4.3. Formulare des ResourceManagers

4.3.1. Ressourcen

Hier erfolgt die Anzeige aller definierten Ressourcen und deren aktueller Zustand.



Resources

Name	Mode	State	Stations
R0		Free	
R1	100	Shared	TMCE System 1
R2		Free	
R3		Free	
R4		Free	
R5		Free	
R6		Free	
R7		Free	

Abbildung 6: Ressourcen

Die Ressourcen im ResourceManager können entweder exklusiv (exclusive) oder geteilt (shared) einer Instanz des TestManagers zugeordnet werden. Wird eine Ressource gemeinsam benutzt, muss sich diese allerdings in einem bestimmten Modus befinden, welcher gemeinsam benutzt wird, da ansonsten ein Ressourcen - Handling nicht notwendig ist.

Die einzelnen Spalten haben dabei folgende Bedeutung:

- Name (Name der Ressource)
- Mode (Betriebsart, wenn es sich um eine gleichzeitig gemeinsam genutzte Ressource handelt (Shared-State))
- State (Aktueller Status der Ressource)
  Folgende Möglichkeiten bestehen:
  - Free
  - Shared
  - Exclusive
- Stations (Auflistung der Stationen, die gerade die Ressource benutzen, durch Semikolon getrennt)

4.3.2. Overview

Diese Anzeige ermöglicht einen Überblick über alle Stationen und deren aktuellen Zustand.

Overview

Station	State	Block	
TMCE System 1	Pass		
TMCE System 2	Pass		
TMCE System 3	Active	B2[S]:R2	
TMCE System 4	Active	B6[S]:R6	

Abbildung 7: Overview

Die einzelnen Spalten haben dabei folgende Bedeutung:

- Station (Name der TestManager - Instanz)
- State (Aktueller Status)
  Folgende Möglichkeiten bestehen:
  - Start
  - Active
  - Waiting
  - Pass
  - Fail
  - Abort
- Block (Ausgeführter Block, wenn die Station aktiv ist)

4.3.3. Stationen

Für jede Instanz des TestManagers wird ein Formular für die entsprechende Station erstellt.

TMCE System 1

Block	Type	Condition	State	Resources	
B0	Changeable		Pass	R0	
B1	Sequential	B0	Pass	R1[100]	
B2	Sequential	B1	Pass	R2[200]	
B3	Changeable		Pass	R3	
B4	Changeable		Pass	R4	
B5	Sequential	B4	Active	R5	
B6	Sequential	B5	Waiting	R6	
B7	Changeable		Waiting	R7	

Abbildung 8: Stationen

Hier werden die Testblöcke der aktuellen Testsequenz abgebildet.

Die einzelnen Spalten haben dabei folgende Bedeutung:

- Block (Name der Blocks in der Sequenz - Liste)
- Type (Art des Blocks, eine genaue Beschreibung der Blockarten erfolgt weiter hinten im Dokument)
  Folgende Blöcke sind möglich:
  - Fix
  - All
  - Changeable
  - Sequential
- Condition (Optionale Bedingung, wann der Block ausgeführt werden darf)
- State (Status des Blocks)
  Folgende Werte sind möglich:
  - Waiting
  - Abort
  - Active
  - Pass
  - Fail
  - Omitted
- Resources (Auflistung aller mit dem Block verknüpften Ressourcen, durch Semikolon getrennt)

4.3.4. History

Hier erfolgt eine chronologische Auflistung, wann welcher Block in welcher Station abgearbeitet wurde oder eine Station auf freie Ressourcen warten musste.

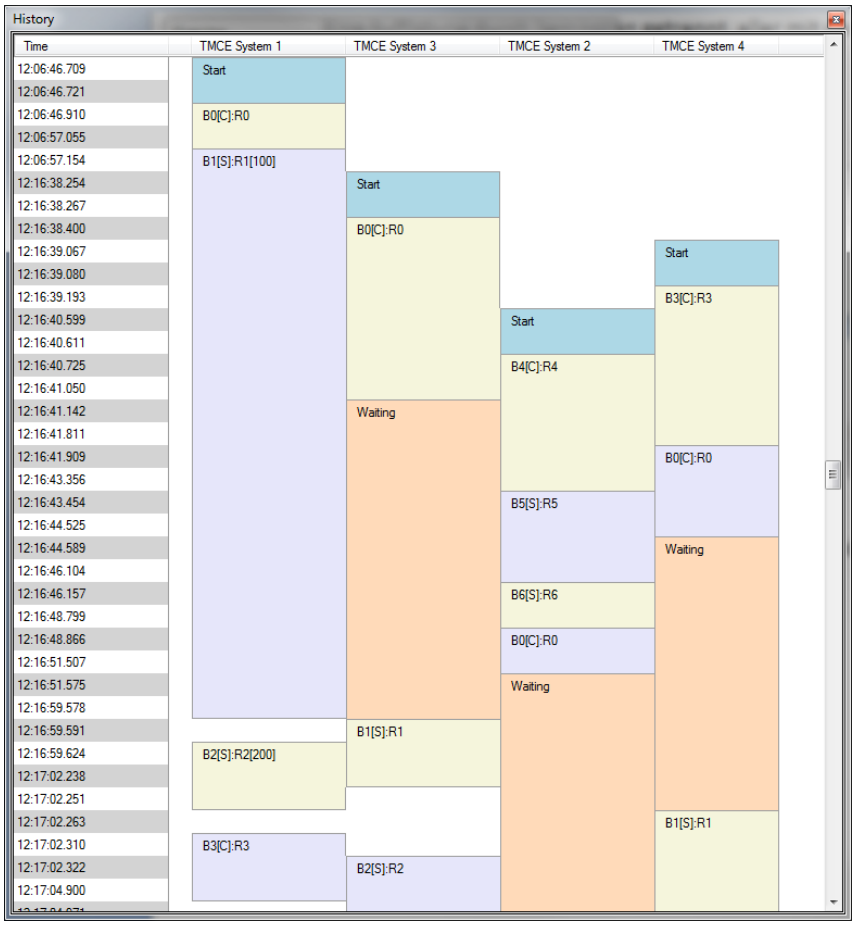


Abbildung 9: History

5. Handbuch Software

5.1. Definition von Abläufen

Im Zusammenspiel mit dem TestManager müssen die definierten Testblöcke einer festgelegten Syntax entsprechen. Diese Syntax hat folgenden Aufbau:

BlockName[BlockType:Condition]:Ressource[Mode]

Dabei sind „BlockType“, „Condition“, „Ressource“ und „Mode“ optional. Nachfolgend werden die bisher definierten Blocktypen beschrieben.

5.1.1. Feste Blöcke (Fix | F)

Ein solcher Ablaufblock definiert einen Block, welcher erst ausgeführt werden darf, wenn alle vorher definierten Blöcke abgearbeitet wurden. Außerdem dürfen Blöcke, welche sich hinter diesem Block befinden, nicht vor diesem Block abgearbeitet werden. Ein solcher Block stellt also einen Fix - Punkt in einem Ablauf dar. Wird in der Block - Definition kein „BlockType“ angegeben, so entspricht dies einem fixen Block.

- Syntax: F oder Fix
- Folgende Beispiele definieren einen fixen Block, welcher nicht an eine Ressource gebunden ist:
  - Block01
  - Block01[F]
  - Block01[Fix]
- Folgendes Beispiel definiert einen fixen Block, welcher an die Ressource „Res01“ gekoppelt ist (Ressource muss dabei exklusiv zur Verfügung stehen):
  - Block01[F]:Res01
- Folgendes Beispiel definiert einen fixen Block, welcher an die Ressource „Res01“ gekoppelt ist (Ressource kann gemeinsam verwendet werden, muss sich aber im Mode „100“ befinden. Ist Ressource bisher frei, wird sie in den Mode „100“ versetzt):
  - Block01[F]:Res01[100]

5.1.2. Feste Blöcke über alle aktiven Stationen (All | A)

Ein solcher Ablaufblock definiert einen Block, welcher erst ausgeführt werden darf, wenn alle vorher definierten Blöcke in allen aktiven Stationen (also Stationen, welche nicht „PASS“, „FAIL“ oder „Disabled“ sind) abgearbeitet wurden. Außerdem dürfen Blöcke, welche sich hinter diesen Block befinden, nicht vor diesem Block abgearbeitet werden.

- Syntax: A oder All
- Folgende Beispiele definieren einen fixen Block, welcher nicht an eine Ressource gebunden ist:
  - Block01[A]
  - Block01[All]
- Folgendes Beispiel definiert einen fixen Block, welcher an die Ressource „Res01“ gekoppelt ist (Ressource muss dabei exklusiv zur Verfügung stehen):
  - Block01[A]:Res01
- Folgendes Beispiel definiert einen fixen Block, welcher an die Ressource „Res01“ gekoppelt ist (Ressource kann gemeinsam verwendet werden, muss sich aber im Mode „100“ befinden. Ist Ressource bisher frei, wird sie in den Mode „100“ versetzt):
  - Block01[A]:Res01[100]

5.1.3. Austauschbare Blöcke (Changeable | C)

Ein solcher Block darf im Ablauf übersprungen und zu einem späteren Zeitpunkt ausgeführt werden, wenn die dafür notwendigen Ressourcen momentan nicht zur Verfügung stehen. Es kann als Bedingung ein Block definiert werden, welcher abgearbeitet werden muss, bevor ein solcher Block ausgeführt werden darf.

- Syntax: C oder Changeable
- Folgende Beispiele definieren einen austauschbaren Block, welcher nicht an eine Ressource gebunden ist:
  - Block01[C]
  - Block01[Changeable]
- Folgendes Beispiel definiert einen austauschbaren Block, welcher an die Ressource „Res01“ gekoppelt ist (Ressource muss dabei exklusiv zur Verfügung stehen):
  - Block01[C]:Res01
- Folgendes Beispiel definiert einen austauschbaren Block, welcher nicht an eine Ressource gekoppelt ist und erst ausgeführt werden darf, wenn „Block01“ abgeschlossen wurde:
  - Block02[C:Block01]
- Folgendes Beispiel definiert einen austauschbaren Block, welcher an die Ressource „Res01“ gekoppelt ist und erst ausgeführt werden darf, wenn „Block01“ abgeschlossen wurde (Ressource muss dabei exklusiv zur Verfügung stehen):
  - Block02[C:Block01]:Res01
- Folgendes Beispiel definiert einen austauschbaren Block, welcher an die Ressource „Res01“ gekoppelt ist (Ressource kann gemeinsam verwendet werden, muss sich aber im Mode „100“ befinden. Ist Ressource bisher frei, wird sie in den Mode „100“ versetzt):
  - Block01[C]:Res01[100]

5.1.4. Sequenz - Blöcke (Sequential | S)

Ein solcher Block muss immer unmittelbar nach dem in der Bedingung (Condition) angegebenen Block ausgeführt werden. Ein solcher Block kann übersprungen werden, wenn auch der in der Bedingung angegebene Block übersprungen wird.

- Syntax: S oder Sequential
- Folgende Beispiele definieren einen Sequenz - Block, welcher nicht an eine Ressource gebunden ist und unmittelbar nach dem davor definierten Block ausgeführt werden muss:
  - Block01[S]
  - Block01[Sequential]
- Folgendes Beispiel definiert einen Sequenz - Block, welcher an die Ressource „Res01“ gekoppelt ist und unmittelbar nach dem davor definierten Block ausgeführt werden muss (Ressource muss dabei exklusiv zur Verfügung stehen):
  - Block01[S]:Res01
- Folgendes Beispiel definiert einen Sequenz - Block, welcher nicht an eine Ressource gekoppelt ist und unmittelbar nach „Block01“ ausgeführt werden muss:
  - Block04[S:Block01]

5.2. Befehle des ResourceManagers

Zur Synchronisation der Ablaufsteuerung mit dem TestManager wurden folgende Befehle im ResourceManager implementiert.

5.2.1. SetBlockList

Mit diesem Befehl wird die Konfiguration der Testsequenz an den ResourceManager gesendet.

- Bsp. (IP_DSTRT)

```
var
  sSystemString : string;
  vsBlocks : stringvector;

step

  vsBlocks := GetSeqBlocks;
  Debug.Show(1, vsBlocks);
  sSystemString := 'TMCE System ' + Str(globalvar.get('TMCE_System'));
  Toolmanager.SetValue('HWCONTROL', 'SetBlockList.' + sSystemString, vsBlocks);

end.
```

5.2.2. ResetBlockList

Mit diesem Befehl wird die Konfiguration der Testsequenz zurückgesetzt.

- Bsp. (IP_RESET)

```
var
  sSystemString : string;

step

  sSystemString := 'TMCE System ' + Str(globalvar.get('TMCE_System'));
  Toolmanager.SetEvent('HWCONTROL', 'ResetBlockList.' + sSystemString);

end. .
```

5.2.3. DisableBlockList

Mit diesem Befehl wird die Station (Instanz des TestManagers) als „nicht aktiv“ gekennzeichnet.

- Bsp. (IP_RESET)

```
var
  sSystemString : string;

step

  sSystemString := 'TMCE System ' + Str(globalvar.get('TMCE_System'));
  if RegForm.Get('ENABLED') = '0' then begin
    Toolmanager.SetEvent('HWCONTROL', 'DisableBlockList.' + sSystemString);
  end else begin
    Toolmanager.SetEvent('HWCONTROL', 'ResetBlockList.' + sSystemString);
  end;

end.
```

5.2.4. GetNextBlock

Mit diesem Befehl wird der nächste Block angefragt, welcher abgearbeitet werden soll.

- Bsp. (IP_BSTRT)

```
var
  sSystemString : string;
  sBlock : string;

step

  if Len(GetSeqBlocks) > 0 then begin

    sSystemString := 'TMCE System ' + Str(globalvar.get('TMCE_System', 0));
    while sbLock = '' do begin
      sBlock := Toolmanager.GetString('HWCONTROL', 'GetNextBlock.' + sSystemString);
      if sbLock = '' then begin
        DateTime.Delay(10, true);
      end;
    end;

    SetSeqBlock(sBlock);

  end;

end.
```

5.2.5. GetNextBlockAndPreferResources

Mit diesem Befehl wird der nächste Block angefragt, welcher abgearbeitet werden soll. Dabei werden Blöcke, welche an Ressourcen gebunden sind, bevorzugt.

- Bsp. (IP_BSTRT)

```
var
  sSystemString : string;
  sBlock : string;

step

  if Len(GetSeqBlocks) > 0 then begin

    sSystemString := 'TMCE System ' + Str(globalvar.get('TMCE_System', 0));
    while sbLock = '' do begin
      sBlock := Toolmanager.GetString('HWCONTROL', GetNextBlockAndPreferResources.' +
                sSystemString);
      if sbLock = '' then begin
        DateTime.Delay(10, true);
      end;
    end;

    SetSeqBlock(sBlock);

  end;

end.
```

5.2.6. SetBlockResult

Mit diesem Befehl wird das Testergebnis nach dem Ende eines Blockes übergeben.

- Bsp. (IP_BEND)

```
var
  sSystemString : string;
  sBlock : string;

step

  if (Len(GetSeqBlocks) > 0) and (GetSeqBlock <> '') then begin

    sSystemString := 'TMCE System ' + Str(globalvar.get('TMCE_System', 0));

    if GetTestResult = 1 then begin
      Toolmanager.SetValue('HWCONTROL', 'SetBlockResult.' + sSystemString, 'Pass:' + GetSeqBlock);
    end else if GetTestResult = 3 then begin
      Toolmanager.SetValue('HWCONTROL', 'SetBlockResult.' + sSystemString, 'Break:' +
                            GetSeqBlock);
    end else begin
      Toolmanager.SetValue('HWCONTROL', 'SetBlockResult.' + sSystemString, 'Fail:' + GetSeqBlock);
    end;

    if SeqBlockCanContinue then begin

      Debug.Show(1, GetRemainingSeqBlocks);

      if (Len(GetRemainingSeqBlocks) > 0) or
         (Toolmanager.GetValue('HWCONTROL', 'AllBlocksFinished.' + sSystemString) = 0) then begin
        SeqBlockContinue;
      end;
    end;

  end;

end.
```

5.2.7. AllBlocksFinished

Mit diesem Befehl kann abgefragt werden, ob alle Blöcke abgearbeitet wurden.

- Bsp. (IP_BEND)

```
var
  sSystemString : string;
  sBlock : string;

step

  if (Len(GetSeqBlocks) > 0) and (GetSeqBlock <> '') then begin

    sSystemString := 'TMCE System ' + Str(globalvar.get('TMCE_System', 0));

    if GetTestResult = 1 then begin
      Toolmanager.SetValue('HWCONTROL', 'SetBlockResult.' + sSystemString, 'Pass:' + GetSeqBlock);
    end else if GetTestResult = 3 then begin
      Toolmanager.SetValue('HWCONTROL', 'SetBlockResult.' + sSystemString, 'Break:' + GetSeqBlock);
    end else begin
      Toolmanager.SetValue('HWCONTROL', 'SetBlockResult.' + sSystemString, 'Fail:' + GetSeqBlock);
    end;

    if SeqBlockCanContinue then begin

      Debug.Show(1, GetRemainingSeqBlocks);

      if (Len(GetRemainingSeqBlocks) > 0) or
         (Toolmanager.GetValue('HWCONTROL', 'AllBlocksFinished.' + sSystemString) = 0) then begin
        SeqBlockContinue;
      end;
    end;

  end;

end.
```

5.2.8. SetTestResult

Mit diesem Befehl wird das Ergebnis des gesamten Tests übergeben.

- Bsp. (IP_DEND)

```
var
  sSystemString : string;

step

  sSystemString := 'TMCE System ' + Str(globalvar.get('TMCE_System', 0));

  if GetTestResult = 1 then begin
    Toolmanager.SetValue('HWCONTROL', 'SetTestResult.' + sSystemString, 'Pass');
  end else if GetTestResult = 3 then begin
    Toolmanager.SetValue('HWCONTROL', 'SetTestResult.' + sSystemString, 'Break');
  end else begin
    Toolmanager.SetValue('HWCONTROL', 'SetTestResult.' + sSystemString, 'Fail');
  end;

end.
```

5.2.9. GetBlock

Mit diesem Befehl kann ein ganz bestimmter Block definiert angefordert werden. Dieser Befehl gibt „1“ oder „true“ zurück, wenn der Block ausgeführt werden kann.

- Bsp. (IP_BCHNG)

```
var
  sSystemString : string;
  sBlock : string;


step

  if Len(GetSeqBlocks) > 0 then begin

    sSystemString := 'TMCE System ' + Str(globalvar.get('TMCE_System', 0));

    if IsSBS then begin
      if Toolmanager.GetValue('HWCONTROL', 'GetBlock.' + sSystemString + '.' + GetSeqBlock) = 0
then begin
        Screen.Dialog.Bitmap(':1');
        Screen.Dialog.Button('');
        Screen.Dialog.Button('');
        Screen.Dialog.SetText('Block change','Wait for resources for block ' + GetSeqBlock);
        Screen.Dialog.Show(false, 1);
      end;
    end;

    while Toolmanager.GetValue('HWCONTROL', 'GetBlock.' + sSystemString + '.' + GetSeqBlock) = 0
do begin
      DateTime.Delay(10, true);
    end;

    Screen.Dialog.Hide;

  end;

end.
```

5.2.10. SetBlock

Mit diesem Befehl kann ein ganz bestimmter Block festgelegt werden, auch wenn die zugeordnete Ressource nicht frei ist. Der Status der Ressource wird gegebenenfalls von „Exclusive“ auf „Shared“ geändert.

- Bsp. (IP_BCHNG)

```
var
  sSystemString : string;
  sBlock : string;


step

  if Len(GetSeqBlocks) > 0 then begin

    sSystemString := 'TMCE System ' + Str(globalvar.get('TMCE_System', 0));

    Toolmanager.SetValue('HWCONTROL', 'SetBlock.' + sSystemString, GetSeqBlock);

  end;

end.
```