

# Handbuch

## **RMOS3-Profiler**

Version:    vorläufig

---

# Inhaltsverzeichnis

Seite

1. Einleitung / Allgemeines .....	4
1.1 Einbindung in RMOS .....	4
1.2 Protokollierung der Rechenzeitverteilung .....	4
1.3 Protokollierung der Taskaktivitäten.....	5
1.4 Ermittlung der Systemparameter .....	5
1.5 Ermittlung der Interruptsperrzeit .....	5
1.6 Meßmethode und Meßgenauigkeit .....	6
1.7 Priorität der Meßtask.....	7
2. Protokollierung der Rechenzeitverteilung unter RMOS mit dem RMOS3-Profiler.....	8
2.1 Starten einer Messung .....	8
2.1.1 Menü.....	8
2.1.2 Starten der Task <i>RProf</i> mit Taskstart.....	8
2.2 Bildschirmausgaben des RMOS3-Profilers .....	10
2.3 Zyklische Messung starten/ausgeben .....	10
2.4 Speicherbedarf.....	11
3. Protokollierung der Taskaktivitäten.....	12
3.1 Starten einer Messung .....	12
3.1.1 Menü.....	12
3.1.2 Starten der Task <i>RPROF</i> durch Taskstart .....	13
3.2 Bildschirmausgaben des RMOS3-Profilers .....	15
3.3 Darstellung der Interrupts .....	16
3.4 Darstellung der SVC-Parameter für RMOS V3.0 .....	16
3.5 Darstellung der SVC-Parameter für RMOS V3.20 .....	17
3.6 Speicherbedarf.....	18
4. Profiler-Programmierschnittstelle .....	19
5. Konfiguration.....	22
5.1 Benötigte SVCs.....	22
5.2 Konfiguration in der Datei PROFCFG.C .....	22
5.2.1 Einstellung der Systemparameter.....	22
5.2.2 Anpassungen an andere Timer-Bausteine .....	22
5.3 Speicherbedarf.....	22
5.4 Der RMOS3-Profiler als nachladbare Task.....	22
5.5 Der RMOS3-Profiler als Task bei der Systemgenerierung .....	23
5.5.1 Konfiguration des RMOS3-Profilers als Task (Aufruf <i>x_prof_init</i> ) .....	23
5.5.2 Konfiguration der Hooks.....	23
6. Lieferumfang und Installation.....	24
7. Fehlermeldungen .....	25

---

8. Abkürzungsverzeichnis.....	26
-------------------------------	----

<i>Projekt/Titel:</i>	RMOS3-Profiler	<i>Filename:</i>	RM3_PROFILER.DOC
<i>Bezeichnung:</i>	Handbuch	<i>Version:</i>	vorläufig
			Seite 3 von 26

---

# 1. Einleitung / Allgemeines

Der RMOS3-Profiler ermittelt

- die Rechenzeitverteilung,
- die Taskaktivitäten und
- einige Systemparameter.

Mit den Ergebnissen der Rechenzeitverteilung kann das System laufzeitmäßig optimiert werden. Die Aufzeichnung der Taskaktivitäten ist für die Fehlersuche und ebenso auch für die Laufzeitoptimierung hilfreich. Weitere Einsatzmöglichkeiten sind das Messen von SVC-Ausführungszeiten und bestimmten Programmteilen. Als Systemparameter werden z.B. die gemessene maximale Timerverzögerung (Interruptsperrzeit), Code-Adressen bei denen es zu der Timerverzögerung gekommen ist, und der Betriebssystemtakt ausgegeben. Mit Hilfe des Debuggers können die ausgegebenen Adressen nach Interruptsperrungen untersucht werden.

Der RMOS3-Profiler kann in folgenden RMOS-Systemen eingesetzt werden:

- RMOS3 V3.0, V3.11 und V3.20

## 1.1 Einbindung in RMOS

Der RMOS3-Profiler kann auf verschiedene Arten in ein RMOS-System integriert werden:

- als nachladbares CLI Kommando
- mit dem Debugger über das Kommando LOADTASK
- als Task bei der Systemgenerierung

Die gewünschte Messung wird über ASCII-Menüs eingestellt und das Meßergebnis als ASCII-Text dargestellt.

Wird der RMOS3-Profiler über den CLI oder den Debugger nachgeladen, so wird eine weitere Task erzeugt. Mit dieser Task kann eine Messung von einer anderen Task aus direkt ohne Menü gestartet bzw. angehalten werden.

Wenn der RMOS3-Profiler als Task bei der Systemgenerierung eingebunden wird, erfolgt der Taskstart vom Debugger oder von einer Anwendertask aus.

## 1.2 Protokollierung der Rechenzeitverteilung

Die Messung kann über den RMOS3-Profiler für eine bestimmte Zeit oder auf unbestimmte Zeit gestartet werden. Bei einem Start auf unbestimmte Zeit kann laufend die Rechenzeitverteilung ausgegeben werden. Es werden die Rechenzeitverteilung für die einzelnen Tasks, die IDLE-Zeit (Leerlaufzeit im System) und die verbrauchte Rechenzeit für die S-Zustände angezeigt.

Die Rechenzeit für die S-Zustände entspricht weitgehend der Rechenzeit der Treiber.

Die Rechenzeit für die weiteren Betriebszustände von RMOS (DI- und I-Zustand) ist in obigen Zeiten enthalten und kann nicht getrennt ermittelt werden.

Die Rechenzeit für die S-Zustände wird wie folgt ermittelt:

$\text{Rechenzeit} = \text{gesamte Meßzeit} - \text{gesamte Rechenzeit für Tasks} - \text{IDLE-Zeit}$
---

Die IDLE-Zeit wird mit der sogenannten Busy Task berechnet, welche mit der niedrigsten Priorität (Priorität 1, da Priorität 0 reserviert ist) im System läuft. Bei den Rechenzeiten für die Tasks sind alle SVC-Ausführungszeiten (Ausnahme: RIO mit Warten) enthalten. Die Rechenzeit für RIO mit Warten ist im S-Zustand enthalten.

Projekt/Titel:	RMOS3-Profiler	Filename:	RM3_PROFILER.DOC
Bezeichnung:	Handbuch	Version:	vorläufig
			Seite 4 von 26

---

Die Berechnung der Rechenzeit ist so ausgelegt, daß eine Messung auch über längere Zeit (für Millisekunden-Ausgaben max. 47 Tage, für Prozent-Ausgaben max. 11,9 Stunden) möglich ist, ohne daß es zu einem Überlauf kommt. Es ist keine Floatingpoint-Unterstützung notwendig.

### 1.3 Protokollierung der Taskaktivitäten

Die Messung kann über den RMOS3-Profiler für eine bestimmte Zeit oder auf unbestimmte Zeit gestartet werden. Bei einem Start auf unbestimmte Zeit können laufend die Taskaktivitäten angezeigt werden. Zu den Taskaktivitäten zählen folgende Ereignisse:

- SVC-Aufrufe
- HW-Interrupt (DI-STATE)
- Betriebssystemzustand infolge eines HW-Interrupts (S-STATE)
- Betriebssystemzustand infolge eines SW-Interrupts (S-STATESW)
- Taskbearbeitung beginnen (TASKIN)
- Taskbearbeitung beenden (TASKOUT)
- E/A-Anforderung (RIO) beenden (RIOEND)

Alle oben genannten Ereignisse außer SVC-Aufrufe werden als Systemzustände bezeichnet. Alle SVC-Aufrufe können einzeln selektiert und optional auf bestimmte Tasks begrenzt werden. Weiterhin kann bei SVCs mit Betriebsmitteln auch eine bestimmte Betriebsmittel-ID ausgewählt werden.

Folgende Meßbeispiele zeigen die verschiedenen Protokollierungsmöglichkeiten der Taskaktivitäten:

- Es sollen alle SVCs "RmSendMail" ("SEND") und "RmReceiveMail" ("RECV") an der Mailbox 3 von den Tasks 5 und 6 aufgezeichnet werden.
- Es sollen alle SVCs aller Tasks und die Systemzustände angezeigt werden.
- Es sollen alle Semaphore-Aufrufe aller Tasks mit der Semaphore-ID 7 aufgezeichnet werden.

Alle Informationen werden in einem von der Task angeforderten Speicherbereich abgelegt. Der Speicherbereich ist als Ringpuffer organisiert, d.h. bei Überlauf gehen die ältesten Informationen verloren. Jede Taskaktivität wird mit einem Zeitstempel (Genauigkeit in µsec) versehen.

### 1.4 Ermittlung der Systemparameter

Folgende Systemparameter werden angezeigt:

- Betriebssystemtakt in ms
- RMOS-Version
- Priorität des RMOS3-Profilers
- Interruptsperrzeit in µsec
- Minimale und maximale Timerverzögerung in µsec
- Die letzten vier Code-Adressen mit der zugehörigen Interruptsperrzeit in µsec, bei deren Abarbeitung sich die Interruptsperrzeit verschlechtert hat
- Triggeradresse für Emulator (ICE)

### 1.5 Ermittlung der Interruptsperrzeit

Die Interruptsperrzeit wird mit dem Betriebssystemtakt ermittelt und ist die Differenz zwischen der minimalen und maximalen Timerverzögerung. Die minimale Timerverzögerung ist die kürzeste Zeit und die maximale Timerverzögerung ist die längste Zeit nach der der Betriebssystemtakt angenommen wurde. Der Verlust von Timerticks aufgrund einer zu langen Interruptsperrzeit kann nicht erkannt werden, ebenso eine Interruptsperrzeit zwischen zwei Timerticks. Die angezeigte Interruptsperrzeit entspricht also nicht unbedingt der tatsächlichen maximalen Interruptsperrzeit. Sofern jedoch keine unverhältnismäßig hohen

Projekt/Titel:	RMOS3-Profiler	Filename:	RM3_PROFILER.DOC
Bezeichnung:	Handbuch	Version:	vorläufig Seite 5 von 26

---

Interruptsperrzeiten auftreten, kann man die gemessene Interruptsperrzeit mit der maximalen Interruptsperrzeit gleichsetzen.

Der Verursacher für die Interruptsperrzeit kann mit den angezeigten Adressen ausfindig gemacht werden. Es werden die letzten vier Code-Adressen mit den zugehörigen Interruptsperrzeiten in µsec dargestellt, bei denen es zu einer Timerverzögerung gekommen ist.

Mit Hilfe des Debugger-Kommandos "ASM" können die angegebenen Adressen untersucht werden. Da der Prozessor bei einem "Interrupt enable" (Befehl STI) noch zwei Befehle ausführt bis tatsächlich die Interrupts wieder freigegeben werden, wird nicht ab der angegebenen Adresse disassembliert, sondern schon früher (z.B. 15 Bytes früher) damit der Befehl "STI" sichtbar wird.

Eine weitere Möglichkeit bietet sich mit der Aufzeichnung der Taskaktivitäten kombiniert mit der Möglichkeit, eine Interruptsperrzeit einzustellen. Mit Erreichen der eingestellten Interruptsperrzeit wird die Protokollierung der Taskaktivitäten angehalten (sofern sie vorher gestartet war). Anhand der Aufzeichnung kann der Verursacher festgestellt werden. Gehen Sie dabei wie folgt vor:

- Aufzeichnung der Taskaktivitäten starten
- Systemparameter ausgeben und Messung zurücksetzen
- Schwellwert für maximale Interruptsperrzeit einstellen
- mit "Systemparameter ausgeben" kontrollieren, ob eingestellte Interruptsperrzeit aufgetreten ist, wenn ja
- Aufzeichnungen der Taskaktivitäten untersuchen (am Schluß). Die Aufzeichnungen wurden beim Erreichen der Interruptsperrzeit beendet.

In folgenden Fällen ist die angezeigte Adresse falsch:

- Wenn unter Interruptsperrung der Befehl IRETD ausgeführt und dabei die Interrupts freigegeben werden und nach dem IRETD sofort ein Timerinterrupt auftritt
- Wenn einem "Interrupt enable" (STI Befehl) ein Unterprogrammaufruf oder ein Sprungbefehl folgt (die Interrupts werden erst nach dem zweiten nachfolgenden Befehl freigegeben)

Wenn die angezeigte Adresse einen ungültigen Selektor enthält, so wurde die Timerverzögerung durch ein nachladbares Programm verursacht, das sich vor der Ausgabe mit dem RMOS3-Profiler beendete.

Weiterhin kann mit einem ICE auf das Auftreten der eingestellten max. Interruptsperrzeit getriggert und mit dem Trace die Ursache untersucht werden.

## 1.6 Meßmethode und Meßgenauigkeit

Für die Messung wird der Timer-Baustein verwendet, der den Betriebssystemtakt erzeugt. Die Genauigkeit bei den Zeitangaben entspricht damit der Eingangsfrequenz des Timer-Bausteins. Bei der Auswertung der Rechenzeitverteilung werden die Counter-Werte berücksichtigt, sofern sie ein Vielfaches einer Millisekunde darstellen (Auswertung nur bis Millisekunden, keine Rundung). Die Genauigkeit ist also unabhängig von der Auflösung des Betriebssystemtakts (SYSTEMCLOCK bzw. X\_PIT\_MS) und beträgt für den PC ca. eine Mikrosekunde.

Nachfolgend sind die durch den RMOS3-Profiler verursachten Overheadzeiten angegeben. Diese Zeiten wurden für einen PC mit Intel 486SX25-Prozessor gemessen. Für schnellere PCs verkürzen sich die Zeiten entsprechend.

- Das Verdrängen oder Aufsetzen einer Task verlängert sich um ca. 25µs.
- Die Ausführungszeit eines SVCs verlängert sich bedingt durch die Hookeinsprünge um ca. 30µs.
- Das Eintreten in den S-Zustand und das Verlassen des S-Zustands verlängert sich um ca. 15µs.
- Die Interruptlatenzzeit erhöht sich bedingt durch das Tracen des Interrupts um ca. 15µs.

Die tatsächlich vom RMOS3-Profiler verursachte Overheadzeit in einer Anwendung hängt in erster Linie von dieser Anwendung ab:

- Wenn von einem System nur wenige (einige zehn) SVCs pro Sekunde abgesetzt werden, dann beträgt die durch den RMOS3-Profiler verursachte Overheadzeit nur etwa ein Prozent.

Projekt/Titel:	RMOS3-Profiler	Filename:	RM3_PROFILER.DOC
Bezeichnung:	Handbuch	Version:	vorläufig
		Seite 6 von 26	

- Liegt hingegen ein System vor, bei dem periodisch jede ms ein Interrupt anliegt, ein Wechsel in den S-Zustand erfolgt, von da aus ein SVC abgesetzt wird, der zu einem Taskwechsel führt, dann kann der RMOS3-Profiler 15% oder mehr der gesamten zur Verfügung stehenden Rechenzeit benötigen.
- Es sind aber auch Anwendungen denkbar, welche fast ausschließlich aus SVCs bestehen. Dann kann die durch den RMOS3-Profiler beanspruchte Rechenzeit 50% und mehr der gesamten Rechenzeit betragen.

## 1.7 Priorität der Meßtask

Wird der RMOS3-Profiler mit dem CLI gestartet, so kann über die Aufrufzeile die Priorität der Task des RMOS3-Profilers angegeben werden. Fehlt in der Aufrufzeile die Priorität bzw. wird der RMOS3-Profiler mit dem Kommando LOADTASK des Debuggers gestartet, so wird die Defaultpriorität 250 verwendet. Die Priorität der Meßtask hat keinen Einfluß auf das Sammeln der Meßdaten. Sie hat lediglich folgende Auswirkungen auf die Ausgabe der Meßergebnisse:

- Die Bildschirmausgabe ist nicht kontinuierlich.
- Bei der Ausgabe der Rechenzeitverteilung stimmt die Summe der Rechenzeiten nicht exakt überein

Beispiel für Start des RMOS3-Profilers über den CLI mit der Priorität 100:

**CLIPROMPT>**RPROF 100 <RETURN>

Hinweis: Die Priorität wird als Dezimalzahl angegeben.

## 2.

Projekt/Titel:	RMOS3-Profiler	Filename:	RM3_PROFILER.DOC
Bezeichnung:	Handbuch	Version:	vorläufig
			Seite 7 von 26

---

## 2. Protokollierung der Rechenzeitverteilung unter RMOS mit dem RMOS3-Profiler

### 2.1 Starten einer Messung

Die Messung kann über ein Menü oder durch Taskstart angestoßen werden. Wird der RMOS3-Profiler mit dem CLI oder über LOADTASK (Debugger) geladen, erscheint am Bildschirm ein Menü. Wenn der RMOS3-Profiler als Task im System eingebunden ist, wird der RMOS3-Profiler durch Taskstart aktiviert. Dabei kann über die Startparameter mit und ohne Menü eine Messung gestartet werden. Nach dem Start des RMOS3-Profiler erscheint folgende Startmeldung (z.B. für RMOS3 V3.20):

RMOS Profiler Vx.y running on RMOS3 Version V3.20.09

#### 2.1.1 Menü

Die Bedienung erfolgt über ein Hauptmenü und verschiedene Untermenüs. Alle Ein-/Ausgaben erfolgen in englischer Sprache. In diesem Handbuch sind neben den englischen Texten die deutschen Texte geschrieben.

Select measurement mode  
0 - Determine load distribution  
1 - Determine task activity  
2 - Activate screen paging  
3 - Deactivate screen paging  
4 - Terminate program  
5 - Show operation parameter  
Input: <0>

Auswahl der Messart  
0 - Rechenzeitverteilung ermitteln  
1 - Taskaktivitäten ermitteln  
2 - Bildschirm-Paging einschalten  
3 - Bildschirm-Paging abschalten  
4 - Programm beenden  
5 - System-Parameter anzeigen  
Eingabe: <0>

Determine load distribution  
0 - Start measurement  
1 - Stop measurement  
2 - Start measurement of specified duration  
3 - Output measurement  
4 - Return to main menu  
5 - Release memory  
6 - Start cyclic measurement  
7 - Output cyclic measurement  
Input:<0>

Rechenzeitverteilung ermitteln  
0 - Messung starten  
1 - Messung stoppen  
2 - Messung mit bestimmter Länge starten  
3 - Rechenzeitverteilung ausgeben  
4 - Zurück zum Hauptmenü  
5 - Speicher freigeben  
6 - Zyklische Messung starten  
7 - Zyklische Messung ausgeben  
Eingabe: <0>

(wenn 2 gewählt wird, kommen drei weitere Abfragen dazu)

Selection of timeunit  
0 - msec\_10  
1 - sec  
2 - min  
3 - abort input  
Input: <2>

Auswahl der Zeiteinheit  
0 - msec\_10  
1 - sec  
2 - min  
3 - Eingabe abbrechen  
Eingabe: <2>

Number of timeunits (1 - 255, 0 = abort input)  
Input: <2>

Anzahl der Zeiteinheiten(1 - 255, 0=Eingabe abbrechen)  
Eingabe: <2>

Press <RETURN> for starting !

Drücke <RETURN> zum Start !

(nach Eingabe von <RETURN> wird die Messung gestartet)

#### 2.1.2 Starten der Task *RProf* mit Taskstart

Der RMOS3-Profiler katalogisiert unter den Namen "RProfStrt" eine Task, die wie eine gewöhnliche Task mit dem Debugger oder von einer anderen Task gestartet werden kann. Der Task werden beim Start über

Projekt/Titel:	RMOS3-Profiler	Filename:	RM3_PROFILER.DOC
Bezeichnung:	Handbuch	Version:	vorläufig
			Seite 8 von 26



die Startparameter EAX und EBX oder durch ein Menü die notwendigen Parameter für die Messung mitgeteilt.

#### Parameter EBX:

EBX enthält die Länge der Messung. Es werden nur die unteren 16 Bits ausgewertet. Die Eingabe erfolgt hexadezimal.

Die Darstellung der Meßzeit entspricht der RMOS-Zeitdarstellung (Zeiteinheit, Anzahl der Zeiteinheiten, siehe auch Headerdatei svc.h).

```
Bit   |15|14|13|12|11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
      -----
                |                               |- Count
                |- Time Unit
```

#### Parameter EAX:

```
Bit   | 8|.5| 4| 3| 2| 1| 0|
      -----
                |                               |- 0: interaktive Abfrage
                |                               | 1: keine interaktive Abfrage
                |                               |- 0: Rechenzeitverteilung ermitteln
                |                               | 1: Taskaktivitäten ermitteln
                |- 000: Messung starten
                | 001: Messung stoppen
                | 010: Messung mit bestimmter Länge starten(EBX)
                | 011: reserviert
                | 100: reserviert
                | 101: Speicher freigeben
                | 110: zyklische Messung starten
                | 111: reserviert
                |- xxxx: Anzahl der zyklischen Messungen
```

Alle weiteren Bits in EAX sind reserviert.

Wenn das Bit für interaktive Abfrage gesetzt ist, werden keine weiteren Bits mehr ausgewertet (alle Parameter werden über das Hauptmenü und die Untermenüs abgefragt). Falls die Messung für eine bestimmte Länge gestartet wurde, muß für weitere Aufträge an die Task das Ende der laufenden Messung abgewartet werden.

Ist eine Messung mit "Start measurement" ("Messung starten") angestoßen worden, kann zu jedem Zeitpunkt die aktuelle Rechenzeitverteilung ausgegeben werden.

Das Beenden der RPofStrt-Task beeinflusst nicht eine laufende Messung oder die bereits ermittelte Rechenzeitverteilung, d.h. die Task kann zwischendurch jederzeit beendet und neu gestartet werden. Ist eine Messung gestartet, so kann jederzeit die aktuelle Rechenzeit ausgegeben werden.

#### Beispiele für Start der Task RPROF mit dem Debugger:

(Die <TASK ID> von RPROF kann mit dem Kommando "DIR" des Debuggers ermittelt werden)

##### Beispiel 1:

Es soll eine Messung für 2 s gestartet werden:      START <TASK ID> 302:9

##### Beispiel 2:

Es soll eine Messung für unbestimmte Zeit gestartet werden:      START <TASK ID> 1

##### Beispiel 3:

Es soll eine Messung gestoppt werden:      START <TASK ID> 5

Projekt/Titel:	RMOS3-Profiler	Filename:	RM3_PROFILER.DOC
Bezeichnung:	Handbuch	Version:	vorläufig    Seite 9 von 26

#### Beispiel 4:

Die Task *RPROF* soll im interaktiven Modus gestartet werden:

START <TASK ID> 0:0  
oder START <TASK ID> 0

Select measurement mode  
0 - Determine load distribution  
1 - Determine task activity  
2 - Activate screen paging  
3 - Deactivate screen paging  
4 - Terminate program  
5 - Show operation parameters  
Input: <0>

Weitere Bedienungshinweise sind dem Kapitel Menü (siehe 2.1.1) zu entnehmen.

## 2.2 Bildschirmausgaben des RMOS3-Profilers

Die Rechenzeitverteilung wird wie folgt am Bildschirm dargestellt:

```
Starttime of measurement:      00:00:02:000 (h:min:s:ms)
Stoptime of measurement:      00:00:03:000 (h:min:s:ms)
Duration of measurement:      .....1000 ms
load distribution for tasks:    .....800 ms ( 80.00%)
load distribution for s-state: .....100 ms ( 10.00%)
Idle time:                    .....100 ms ( 10.00%)
```

```
Task computing load      ID  TASK
.....60 ms ( 6.00%)    ...5  TEST_TASK1
.....40 ms ( 4.00%)    ...6  TEST_TASK2
.....600 ms (60.00%)   ...8  TEST_TASK3
.....100 ms (10.00%)   ..10*
```

Der Zeichenstring nach Task-ID wird nur ausgegeben, wenn die Task im symbolischen Directory **zum Zeitpunkt der Ausgabe** katalogisiert ist.

Alle Ausgaben sind dezimal (Ausnahme: Task-ID ist hexadezimal).

Hat die Rechenzeit für eine Task den Wert 0, so war die Task < 1 ms aktiv (Counter-Werte kleiner einer Millisekunde werden nicht ausgegeben, keine Rundung!).

Ein Stern "\*" nach der Task-ID bedeutet, daß diese Task während der Messung mehrfach gelöscht und mit der gleichen ID wieder erzeugt wurde. In diesem Fall ist die ausgegebene Rechenzeit die Summe aller Rechenzeiten der Tasks, die mit dieser ID aktiv waren.

Die Ausgabe am Bildschirm kann über das Hauptmenü (Deactivate/Activate screen paging) so eingestellt werden, daß nach jeder Bildschirmseite mit einem <RETURN> die nächste Bildschirmseite angezeigt wird. Es wird von einer darstellbaren Zeilenanzahl von 24 Zeilen ausgegangen. Bei Eingabe der Zeichenfolge <A> und <RETURN> wird die Ausgabe abgebrochen.

#### Anmerkung:

Wenn die Messung während einer Ausgabe noch läuft, stimmt die Summe der Rechenzeiten der einzelnen Tasks unter Umständen nicht mit der Ausgabe "Load distribution for tasks" überein, da die Rechenzeiten der Tasks während der Ausgabe nach wie vor aktualisiert werden.

## 2.3 Zyklische Messung starten/ausgeben

Eine Sonderform der "Rechenzeitverteilung ermitteln", ist eine zyklische Messung starten. Über das Menü kann eine zyklische Messung gestartet und ausgegeben werden. Es werden die Zeitunit und die Anzahl der Zeitunits für eine Messung ausgewählt. Anschließend wird die Anzahl der Messungen festgelegt. Der Speicherbedarf zum Abspeichern der Messungen wird aus einem Speicherpool entnommen. Die zyklische Messung kann nur durchgeführt werden, wenn keine "normale" Messung läuft. Der Speicher zum Abspeichern der Messung wird vom Speicherpool angefordert und bei der nächsten Messung

Projekt/Titel:	RMOS3-Profiler	Filename:	RM3_PROFILER.DOC
Bezeichnung:	Handbuch	Version:	vorläufig
			Seite 10 von 26

zurückgegeben und neu angefordert. Mit dem Menüpunkt "Release memory" ("Speicher freigeben") wird der Speicher sofort freigegeben.

**Anmerkung:**

Nach dem Start einer zyklischen Messung wird bei "Output measurement" ("Rechenzeitverteilung ausgeben") die letzte zyklische Messung angezeigt.

Beispiel für zyklische Messung:

Select measurement mode

- 0 - Determine load distribution
  - 1 - Determine task activity
  - 2 - Activate screen paging
  - 3 - Deactivate screen paging
  - 4 - Terminate program
  - 5 - Show operation parameters
- Input: <0>

Auswahl der Meßart

- 0 - Rechenzeitverteilung ermitteln
  - 1 - Taskaktivitäten ermitteln
  - 2 - Bildschirm-Paging einschalten
  - 3 - Bildschirm-Paging abschalten
  - 4 - Programm beenden
  - 5 - Systemparameter anzeigen
- Eingabe: <0>

Determine load distribution

- 0 - Start measurement
  - 1 - Stop measurement
  - 2 - Start measurement of specified duration
  - 3 - Output measurement
  - 4 - Return to main menu
  - 5 - Release memory
  - 6 - Start cyclic measurement
  - 7 - Output cyclic measurement
- Input: <6>

Rechenzeitverteilung ermitteln

- 0 - Messung starten
  - 1 - Messung stoppen
  - 2 - Messung mit bestimmter Länge starten
  - 3 - Rechenzeitverteilung ausgeben
  - 4 - Zurück zu Hauptmenü
  - 5 - Speicher freigeben
  - 6 - Zyklische Messung starten
  - 7 - Zyklische Messung ausgeben
- Eingabe: <6>

Selection of timeunit

- 0 - msec\_10
  - 1 - sec
  - 2 - min
  - 3 - abort input
- Input: <1>

Auswahl der Zeiteinheit

- 0 - msec\_10
  - 1 - sec
  - 2 - min
  - 3 - Eingabe abbrechen
- Eingabe: <1>

Number of timeunits (1 - 255, 0 = abort input)  
Input: <2>

Anzahl der Zeiteinheiten (1 - 255, 0=Eingabe abbrechen)  
Eingabe: <2>

Number of cycles (1 - <max\_cycles>, 0 = abort input)  
Input: <10>

Anzahl der Zyklen (1 - <max\_cycles>, 0=abbrechen)  
Eingabe: <10>

Press <RETURN> for starting !

Drücke <RETURN> zum Starten !

(nach Eingabe von <RETURN> wird die Messung gestartet)

Die maximal mögliche Anzahl der Messungen <max\_cycles> wird anhand des größten verfügbaren Speichersegments im Speicherpool ermittelt.

## 2.4 Speicherbedarf

Zum Abspeichern der Rechenzeiten für die einzelnen Tasks werden pro Task neun Bytes (32-Bit für verbrauchte Rechenzeit in ms, 16-Bit für Counter-Wert des Timers, 16-Bit für Task-ID und 8-Bit zur Erkennung, ob Task-ID mehrfach erzeugt wurde) aus dem Speicherpool angefordert. Die maximal mögliche Taskanzahl wird in der Datei PROFCFG.C festgelegt und entspricht der Anzahl der SMRs. Zum Abspeichern der Daten bei einer zyklischen Messung wird vom Speicherpool zusätzlicher Speicher angefordert (maximal mögliche Taskanzahl \* 9 \* Anzahl der Messungen).

Der für die Ermittlung der Rechenzeitverteilung angeforderte Speicher kann mit dem Menüpunkt "Release memory" ("Speicher freigeben") freigegeben werden.

Projekt/Titel:	RMOS3-Profiler	Filename:	RM3_PROFILER.DOC
Bezeichnung:	Handbuch	Version:	vorläufig
			Seite 11 von 26

---

## 3. Protokollierung der Taskaktivitäten

### 3.1 Starten einer Messung

Die Messung kann über ein Menü oder durch Taskstart angestoßen werden. Wird der RMOS3-Profiler mit dem CLI oder über LOADTASK (Debugger) geladen, erscheint am Bildschirm ein Menü.

Wenn der RMOS3-Profiler als Task im System eingebunden ist, wird der RMOS3-Profiler durch Taskstart aktiviert. Dabei kann über die Startparameter mit oder ohne Menü eine Messung gestartet werden.

Nach dem Start des RMOS3-Profilers erscheint die Startmeldung.

#### 3.1.1 Menü

Es gibt ein Hauptmenü und verschiedene Untermenüs.

Select measurement mode  
0 - Determine load distribution  
1 - Determine task activity  
2 - Activate screen paging  
3 - Deactivate screen paging  
4 - Terminate program  
5 - Show operation parameters  
Input: <1>

Auswahl der Meßart  
0 - Rechenzeitverteilung ermitteln  
1 - Taskaktivitäten ermitteln  
2 - Bildschirm-Paging einschalten  
3 - Bildschirm-Paging abschalten  
4 - Programm beenden  
5 - Systemparameter anzeigen  
Eingabe: <1>

Determine task activity  
0 - Start measurement  
1 - Stop measurement  
2 - Start measurement of specified duration  
3 - Output measurement  
4 - Return to main menu  
5 - Release memory  
6 - Output measurement to file RPROF.SAV  
Input: <0>

Taskaktivitäten ermitteln  
0 - Messung starten  
1 - Messung stoppen  
2 - Messung mit bestimmter Länge starten  
3 - Taskaktivitäten ausgeben  
4 - Zurück zu Hauptmenü  
5 - Speicher freigeben  
6 - Taskaktivitäten in RPROF.SAV speichern  
Eingabe: <0>

(wenn 2 gewählt wird, kommen zwei weitere Abfragen dazu)

Selection of timeunit  
0 - msec\_10  
1 - sec  
2 - min  
3 - abort input  
Input: <1>

Auswahl der Zeitunit  
0 - msec\_10  
1 - sec  
2 - min  
3 - Eingabe abbrechen  
Eingabe: <1>

Number of timeunits (1 - 255, 0 = abort input)  
Input: <2>

Anzahl der Zeitunits (1 - 255, 0=Eingabe abbrechen)  
Eingabe: <2>

(wenn "Start of measurement" oder "Start measurement of specified duration" gewählt wird, geht es weiter mit folgender Abfrage:)

SVC selection  
0 - All SVCs with special states (TASKIN, TASKOUT...)  
1 - All SVCs without special states (TASKIN, TASKOUT...)  
2 - Only specified SVCs  
3 - Old SVC selection  
4 - Abort input  
Input: <1>

---

Mit "special states" werden die Systemzustände bezeichnet. Bei "Old SVC selection" wird die vorhergehende Einstellung der Taskaktivitäten bzw. beim ersten Start die Defaulteinstellung übernommen. Mit der Defaulteinstellung werden alle SVCs aller Tasks und die Systemzustände aufgezeichnet.

Wenn 2 gewählt wird, kommen weitere Abfragen dazu:

Mit der nächsten Eingabe wird pro Zeile ein SVC-Name eingegeben.

für RMOS3 V3.0:

Input SVC name e.g. HALOC <RETURN> (t = terminate input):

für RMOS3 V3.20:

Input SVC name e.g. RmAlloc <RETURN> (t = terminate input):

Der SVC-Name kann mit Klein- oder Großbuchstaben eingegeben werden. Neben den SVC-Namen gibt es noch die Systemzustände TASKOUT, TASKIN, RIOEND, S-STATE, S-STATESW und DI-STATE. Die Eingaben können durch <t> abgeschlossen werden.

Es können auch bestimmte SVCs ausgewählt werden.

Wenn nur SVCs ausgewählt, die alle auf dem gleichen Betriebsmittel (z.B. Eventflag) operieren, kann mit der nächsten Abfrage die ID eines bestimmten Betriebsmittels festgelegt werden. Wurden z.B. als SVCs sef, wef und tef ausgewählt, so erscheint die nachfolgende Menüzeile. Wenn z.B. rio und tef ausgewählt werden, so erscheint die nachfolgende Menüzeile nicht.

Selection of OS kind ID (hex., 0ffff = all IDs):

Mit der nächsten Abfrage können die SVCs auf eine bestimmte Task begrenzt werden (maximal auf fünf Tasks).

Selection of task ID (hex., max. 5, 0ffff = all tasks):

Nach der Eingabe der Task-ID kommt die Abfrage über die Größe des Speichers zur Aufzeichnung der Taskaktivitäten.

Buffer size

0 - 1 kByte

1 - 4 kByte

2 - 8 kByte

3 - 16 kByte

4 - 32 kByte

5 - 64 kByte

6 - Special buffer size

7 - Abort input

Input: <2>

Mit Menüpunkt 6 kann jede beliebige Buffergröße in kByte eingegeben werden.

### 3.1.2 Starten der Task *RPROF* durch Taskstart

Der RMOS3-Profiler katalogisiert unter den Namen "RProfStrt" eine Task, die wie eine gewöhnliche Task mit dem Debugger oder von einer anderen Task gestartet werden kann. Der Task *RPROF* werden beim Start über die Startparameter EAX und EBX die notwendigen Parameter für die Messung mitgeteilt.

#### Parameter EBX:

EBX enthält die Länge der Messung. Es werden nur die unteren 16 Bits ausgewertet. Die Eingabe ist hexadezimal. Die Darstellung der Meßzeit entspricht der RMOS-Zeitdarstellung (Zeiteinheiten (timeunits), Anzahl Zeiteinheiten, siehe svc.h).

Projekt/Titel:	RMOS3-Profiler	Filename:	RM3_PROFILER.DOC
Bezeichnung:	Handbuch	Version:	vorläufig
			Seite 13 von 26

---

```

Bit   |15|14|13|12|11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
      -----
                |                               |- Count
                |- Time Unit

```

#### Parameter EAX:

```

Bit   | 7| 6| 5| 4| 3| 2| 1| 0|
      -----
                |                               |- 0: interaktive Abfrage
                |                               | 1: keine interaktive Abfrage
                |                               |- 0: Rechenzeitverteilung ermitteln
                |                               | 1: Taskaktivitäten ermitteln
                |                               |- 000: Messung starten
                |                               | 001: Messung stoppen
                |                               | 010: Messung mit bestimmter Länge starten
                |                               | (EBX)
                |                               | 011: reserviert
                |                               | 100: reserviert
                |                               | 101: Speicher freigeben
- 000: 1k Speicher
  001: 4k Speicher
  010: 8k Speicher
  011: 16k Speicher
  100: 32k Speicher
  101: 64k Speicher

```

Alle weiteren Bits in EAX sind reserviert.

Wenn das Bit für interaktive Abfrage gesetzt ist, werden keine weiteren Bits mehr ausgewertet (alle Parameter werden über das Hauptmenü und die Untermenüs abgefragt). Die Bits 5 bis 7 legen die Größe des angeforderten Speicherbereichs fest.

Es werden alle SVCs und alle Systemzustände aufgezeichnet.

Falls die Messung für eine bestimmte Länge gestartet wurde, muß für weitere Aufträge an die Task das Ende der laufenden Messung abgewartet werden.

Das Beenden der Task beeinflusst nicht eine laufende Messung oder die bereits ermittelten Taskaktivitäten, d.h. die Task kann zwischendurch jederzeit beendet und neu gestartet werden.

#### **Beispiele für Start der Task RPROF mit dem Debugger:**

##### Beispiel 1:

Es soll eine Messung für 20 ms gestartet werden:      START <TASK ID> 102:0B

##### Beispiel 2:

Start der Task *RPROF* im interaktiven Modus:      START <TASK ID> 0:0 oder START <TASK ID> 0

Select measurement mode  
 0 - Determine load distribution  
 1 - Determine task activity  
 2 - Activate screen paging  
 3 - Deactivate screen paging  
 4 - Terminate program  
 5 - Show operation parameters  
 Input: <1>

Weitere Bedienungshinweise sind dem Kapitel Menü (siehe 3.1.1) zu entnehmen.

## 3.2 Bildschirmausgaben des RMOS3-Profilers

Die Taskaktivitäten werden wie folgt am Bildschirm dargestellt:

Duration of measurement: .....10 ms  
 Buffer size(x Bytes) was (in)sufficient

Zeit [ms]	Counter [usec]	Action	TASK ID	ID	
.....0	..0	TASKIN	...5		TEST_TASK1
.....0	120	RSF	...5	...A	TEST_SEM1
.....0	600	TSF	...5	...B	TEST_TASK1
.....0	650	TASKOUT	...5		TEST_TASK1
.....0	670	TASKIN	...6		TEST_TASK2
.....0	800	RIO	...6	...0	BYT_DRIVER
.....0	950	TASKOUT	...6		TEST_TASK2
.....0	990	TASKIN	...7		TEST_TASK3
.....1	100	HALOC	...7	...0	HEAP_POOL
.....1	800	SEND	...7	..10	MBX_3
.....1	910	REF	...7	...6	
.....1	920	DI-STATE	...4	60	Timer Click
.....1	950	DI-STATE		61	Keyboard
.....1	970	S-STATE			
.....2	220	RIOEND	...6	...0	BYT_DRIVER
.....2	280	TASKOUT	...7		TEST_TASK2

Die zuletzt ausgegebenen Meldungen gehören zu den jüngsten Aktionen. Der Zeichenstring nach ID wird nur ausgegeben, wenn das Betriebsmittel oder die aufrufende Task im symbolischen Directory **zum Zeitpunkt der Ausgabe** katalogisiert sind. Es wird zuerst mit der Betriebsmittel-ID und dann mit der Task-ID gesucht. Der erste Parameter beim Betriebssystemtakt ("Timer Click") zeigt die Timerverzögerung (Interruptsperrzeit) in µsec an.

Die Ausgabe am Bildschirm kann über das Hauptmenü (Activate/Deactivate screen paging) so eingestellt werden, daß nach jeder Bildschirmseite mit einem <RETURN> die nächste Bildschirmseite angezeigt wird. Bei Eingabe der Zeichenfolge <A> und <RETURN> wird die Ausgabe abgebrochen.

Mit dem Menüpunkt "Output measurement to file RPROF.SAV" (Taskaktivitäten in Datei RPROF.SAV speichern) werden die Taskaktivitäten in der Datei RPROF.SAV im aktuellen Verzeichnis abgespeichert.

### Anmerkung:

Während der Ausgabe werden keine Taskaktivitäten mehr aufgezeichnet. Weiterhin werden grundsätzlich keine SVCs der Meßtask *RPROF* aufgezeichnet. Nach Beendigung der Ausgaben wird die Aufzeichnung der Taskaktivitäten fortgesetzt.

### 3.3 Darstellung der Interrupts

Um die HW-Interrupts zu erkennen, werden diese nach dem Start des RMOS3-Profilers (oder x\_prof\_init()) bis zu dessen Beendigung durch spezielle Routinen ausgetauscht. Während dem Betrieb des RMOS3-Profilers dürfen also keine HW-Interrupts installiert werden. Alle HW-Interrupts im DI-Zustand (DI-STATE) werden mit der zugehörigen Interruptnummer unter "DI-STATE" abgespeichert. Jeder HW-Interrupt ist wie folgt im symbolischen Directory unter der Kennung "RM\_CATALOG\_MISC" ("CTY\_USER") eingetragen. Die Namen können in der Datei PROFCFG.C geändert werden.

```
"Timer Click"
"Keyboard"
"Second 8259A"
"COM2:", "COM1:"
"IRQ5"
"Floppy Disk"
"LPT1:"
"IRQ8", "IRQ9", "IRQ10", "IRQ11", "IRQ12", "IRQ13"
"Fixed Disk"
"IRQ15"
```

Der RMOS-Timerinterrupt, der den Betriebssystemtakt erzeugt, wird nur dann im S-Zustand abgearbeitet, wenn eines der folgenden Ereignisse eintritt:

- Ablauf eines Timeouts
- Ablauf des Round-Robin Counters

In allen anderen Fällen wird der Timerinterrupt nur im DI-Zustand abgearbeitet (Laufzeit-Optimierung).

### 3.4 Darstellung der SVC-Parameter für RMOS V3.0

Der erste SVC-Parameter gibt die Task-ID der aufrufenden Task an. Der zweite SVC-Parameter hat für die einzelnen SVCs eine individuelle Bedeutung.

ALOC	Pool-ID
BOUND	Mailbox-ID
CATALOG	Betriebsmittel-ID, die katalogisiert wird
CHANGEDESC	Selektor
CHANGEDESCACCES	Selektor
CNTRL	Programm-ID
CPRI	Task-ID der Task, bei der die Priorität geändert wird
CREATDRIV	Treiber-ID
DELDESC	Selektor
DELETEDRIV	Treiber-ID
DELSK	Task-ID der zu löschenden Task
DSCRTE	Diskrete Byte-ID
GETSTAT	Task-ID
HALOC	Pool-ID
HDALOC	Parameter mode
INTRHAND	Interruptvektor
KILLTSK	Task-ID
LIST	Parameter type
LOCKS	Parameter type
LOOK	Parameter wait_type
QSTRT	Task-ID der zu startenden Task
RECV	Mailbox-ID
REF	Flaggruppen-ID
RESBLK	Parameter mode
RESUME	Task-ID der Task, die in den Zustand "BEREIT" überführt wird
RIO	Geräte-ID
RSF	Semaphore-ID

Projekt/Titel:	RMOS3-Profiler	Filename:	RM3_PROFILER.DOC
Bezeichnung:	Handbuch	Version:	vorläufig
			Seite 16 von 26



---

SEF	Flaggruppen-ID
SEFET	Flaggruppen-ID
SEND	Mailbox-ID
SETINT	Interruptvektor
STIME	Parameter fmt_type
STRT	Task-ID der zu startenden Task
TEF	Flaggruppen-ID
TIME	Parameter fmt_type
TIMEOUT	Parameter type
TIMERSTART	Mailbox-ID
TSF	Semaphore-ID
WEF	Flaggruppen-ID
XTENSION	SVC-Subcode

### 3.5 Darstellung der SVC-Parameter für RMOS V3.20

Der erste SVC-Parameter gibt die Task-ID der aufrufenden Task an. Der zweite SVC-Parameter hat für die einzelnen SVCs eine individuelle Bedeutung.

RmActivateTask	Task-ID
RmAlloc	Parameter mode
RmCatalog	Betriebsmittel-ID, die katalogisiert wird
RmChangeDescriptor	Selektor
RmChangeDescriptorAccess	Selektor
RmCreateDriver	Treiber-ID
RmCreateMessageQueue	Task-ID
RmDecode	Parameter Code
RmDeleteBinSemaphore	Semaphore-ID
RmDeleteDescriptor	Selektor
RmDeleteDriver	Treiber-ID
RmDeleteFlagGrp	Flaggruppen-ID
RmDeleteMailbox	Mailbox-ID
RmDeleteMemPool	Pool-ID
RmDeleteMessageQueue	Task-ID
RmDeleteTask	Task-ID der zu löschenden Task
RmDeleteUnit	Treiber-ID
RmFreeAll	Task-ID
RmGetBinSemaphore	Semaphore-ID
RmGetFlag	Flaggruppen-ID
RmGetName	Betriebsmittel-ID
RmGetTaskInfo	Task-ID
RmGetTaskPriority	Task-ID
RmGetTaskState	Task-ID
RmGetTCBAddress	Task-ID
RmIncreaseSMRBound	Anzahl SMR
RmIntrhand	Interruptvektor
RmIO	Geräte-ID
RmKillTask	Task-ID
RmList	Parameter Type
RmMemPoolAlloc	Pool-ID
RmQueueStartTask	Task-ID der zu startenden Task
RmReAlloc	Parameter Mode
RmReleaseBinSemaphore	Semaphore-ID
RmResetFlag	Flaggruppen-ID
RmResetLocalFlag	Task-ID
RmReceiveMail	Mailbox-ID
RmRestartTask	Parameter mode
RmResumeDriver	Treiber-ID
RmResumeTask	Task-ID der Task die in den Zustand "BEREIT" überführt wird

---

RmSendMail	Mailbox-ID
RmSendMessageQueue	Task-ID
RmSendMailDelayed	Mailbox-ID
RmSetDeviceHandler	Interruptvektor
RmSetFlag	Flaggruppen-ID
RmSetFlagDelayed	Flaggruppen-ID
RmSetLocalFlag	Task-ID
RmSetMailboxSize	Mailbox-ID
RmSetMessageQueueSize	Task-ID
RmSetSVC	SVC-Nummer
RmSetTaskPriority	Task-ID der Task, bei der die Priorität geändert wird
RmStartTask	Task-ID der zu startenden Task
RmSuspendDriver	Treiber-ID
RmSuspendTask	Task-ID

### 3.6 Speicherbedarf

Je Aktion werden im Speicher 11 Bytes belegt. Der Speicher wird von einem Speicherpool angefordert und bei der nächsten Messung zurückgegeben und neu angefordert, falls die Speichergröße für den Puffer sich geändert hat.

Der für die Aufzeichnung der Taskaktivitäten angeforderte Speicher kann mit dem Menüpunkt "Release memory" freigegeben werden.

## 4. Profiler-Programmierschnittstelle

Mit der Programmierschnittstelle können die Rechenzeitverteilung und die Taskaktivitäten gestartet und die bereitgestellten Daten von einer Anwender-Task unter RMOS3 aus ausgewertet werden. Ebenso stehen die Systemparameter zur Verfügung.

Die Kommunikation mit dem RMOS3-Profiler erfolgt über zwei Mailboxen und einer Task "RProfMbox". Die Mailboxen und die Task werden beim Initialisieren des RMOS3-Profilers (x\_prof\_init) bzw. beim Starten des RMOS3-Profilers (über den CLI) automatisch erzeugt. Der RMOS3-Profiler stellt an der Mailbox "RProfRecv" eine Botschaft bereit, die auf eine Datenstruktur zeigt. Mit dieser Datenstruktur werden Aufträge und auch die Ergebnisse zwischen der Anwender-Task und dem RMOS3-Profiler ausgetauscht. Die Kommunikations-Task "RProfMbox" läuft unabhängig des RMOS3-Profilers, der über den CLI gestartet wurde. Zu beachten ist jedoch, daß nur **eine von beiden Tasks zu einem Zeitpunkt aktiv** sein darf.

Die Anwender-Task entnimmt mit dem SVC RECV aus der Mailbox "RProfRecv" den Zeiger auf die Datenstruktur, beschreibt bestimmte Felder, um die gewünschte Messung zu starten und schickt die Datenstruktur an die Mailbox "RProfSend" mit dem SVC SEND. Anschließend wird an der Mailbox "RProfRecv" gewartet, bis die Datenstruktur mit dem Ergebnis bereit steht. Alle benötigten Datenfelder bzw. Speicher werden vom RMOS3-Profiler bereitgestellt.

Hinweis:

Während der Abarbeitung eines Auftrages für die Aufzeichnung der Taskaktivitäten wird die Aufzeichnung unterbrochen.

Die Datenstruktur ist im Headerfile PROF.H unter "MESSCOM" definiert.

```
typedef struct messcom_struct
{
    unsigned short kind;           /* select kind of measurement */
    unsigned short mode;          /* select mode of measurement */
    unsigned short pri;           /* priority of NovaProf task */
    unsigned short timeunit;       /* only for specified duration */
    unsigned short timecount;      /* only for specified duration */
    unsigned short cyclecount;     /* reserved */
    unsigned long bufmax;          /* buffer size for task activity in bytes */
    unsigned short masktask[5];    /* task id's for task activity or -1 for none */
    unsigned short maskid;         /* os kind id or -1 for none */
    unsigned char svctyp[SVCMAX];  /* svc selection */
    unsigned short error;          /* status */
    unsigned short version;        /* RMOS version */
    unsigned short pit_ms;         /* timer tick in ms */
    unsigned short pit_count;      /* init count for the clock */
    unsigned short int_latency;    /* interrupt latency */
    unsigned long result_times_l;  /* reserved */
    unsigned long result_action_l; /* reserved */
    RESULT_T *result_times_p;      /* result pointer of load distribution */
    RESULT_A *result_action_p;     /* result pointer of task activity */
} MESSCOM;
```

Die Parameter *kind* bis *svctyp* sind Eingangsparameter, die restlichen Parameter sind Ergebnisparameter.

Mit *kind* wird die Meßart ausgewählt. Folgende #defines sind möglich:

#define PF_LOAD_DISTRIBUTION	für Rechenzeitverteilung ermitteln
#define PF_TASK_ACTIVITY	für Taskaktivitäten ermitteln
#define PF_TERMINATE_PROG	für NovaProf beenden
#define PF_SHOW_OPERATION_PAR	für Systemparameter ausgeben

---

Der Parameter *mode* kann folgende Werte annehmen:

```
#define PF_START_MEAS           für Messung starten
#define PF_STOP_MEAS           für Messung stoppen
#define PF_START_MEAS_INTERVALL für Messung mit bestimmter Länge starten
#define PF_OUTPUT_MEAS         für Messung ausgeben
#define PF_RELEASE_MEM         für Speicher freigeben
#define PF_OUTPUT_MEAS_TO_FILE für Taskaktivitäten in File RPROF.SAV schreiben
```

Der Parameter *pri* legt die Priorität der Task "RProfMbox" fest und kann Werte zwischen 3 und 255 annehmen.

Die Parameter *timeunit* (mögliche Werte siehe *svc.h*) und *timecount* (Werte 1 bis 255) sind nur für "Messung mit bestimmter Länge" notwendig.

Der Parameter *bufmax* ist für "Taskaktivitäten ermitteln" und legt die Größe des angeforderten Speichers in Bytes fest.

Mit *masktask* kann die Aufzeichnung der SVCs auf maximal fünf Tasks begrenzt werden. Das Feld ist mit -1 vorbesetzt, was bedeutet, daß die SVCs von allen Tasks aufgezeichnet werden.

Der Parameter *maskid* ist optional (mit -1 vorbesetzt) und begrenzt die SVC-Aufzeichnung auf eine bestimmte Betriebsmittel-ID.

Mit dem Feld *svctyp* können einzelne SVCs und Systemzustände selektiert werden. Das Feld *svctyp* ist mit 1 vorbesetzt. Damit werden defaultmäßig alle SVCs und Systemzustände aufgezeichnet. Der Index entspricht der SVC-Nummer.

Der Ergebnisparameter *error* teilt dem Anwender mit, ob ein Fehler aufgetreten ist. Nur wenn dieser Parameter 0 ist, liegt ein Ergebnis vor. Mögliche Fehler sind:

```
#define PF_OUT_OF_MEMORY
#define PF_MEAS_NOT_STARTED
#define PF_MEAS_NOT_STOPPED
#define PF_DURATION_NOT_ENOUGH
#define PF_WRONG_INPUT
#define PF_FILE_ERROR
#define PF_TASK_NUM_NOT_ENOUGH
```

Die Parameter *version*, *pit\_ms*, *pit\_count* und *int\_latency* beinhalten die Systemparameter.

*result\_times\_p* und *result\_action\_p* sind Zeiger auf Strukturen, die die Rechenzeitverteilung bzw. die Taskaktivitäten beinhalten (siehe PROF.H).

```
typedef struct result_t_struct /* result of load distribution */
{
    MESSTIME times;
    TASKTIME task_times[1];
} RESULT_T;
```

```
typedef struct result_a_struct /* result of task activity */
{
    MESSACT time_action;
    ACTION task_action[1];
} RESULT_A;
```

---

```

typedef struct messtime_struct
{
    unsigned short task_max;          /* max. task number */
    unsigned short reserved;          /* reserved */
    unsigned long start_msec;          /* starttime in ms */
    unsigned long stopp_msec;          /* stopptime in ms */
    unsigned long dauer_msec;          /* duration in ms */
    unsigned long task_all_msec;        /* task load */
    unsigned long idle_msec;           /* idle time */
    unsigned long stime_msec;           /* s-state time */
    unsigned long corr_msec;           /* reserved */
} MESSTIME;

typedef struct tasktime_struct
{
    unsigned long msec;                /* time in ms */
    unsigned short counter;            /* pic counter */
    unsigned char delete;              /* indicator if a task was deleted */
    unsigned short taskid;             /* task id */
} TASKTIME;

typedef struct action_struct
{
    unsigned char svcnr;               /* svc number */
    unsigned long msec;                /* time in ms */
    unsigned short counter;            /* pic counter */
    unsigned short task_id;            /* task id */
    unsigned short id;                 /* os kind id */
} ACTION;

typedef struct messact_struct
{
    unsigned long dauer_msec;          /* duration in ms */
    unsigned short buf_overnrun;        /* buffer overrun */
    unsigned short index;              /* buffer index */
} MESSACT;

```

---

## 5. Konfigurierung

### 5.1 Benötigte SVCs

Folgende SVCs werden benötigt:

catalog, cpri, create, createdesc, deldesc, endt, intrhand, getsize, haloc, hdaloc, killtsk, list, locks, look, pause, physadr, strt, uncatalog und xtension.

### 5.2 Konfiguration in der Datei PROFCFG.C

Nach den Anpassungen in der Datei PROFCFG.C ist diese zu übersetzen und beim RMOS3-Profiler als Task bei der Systemgenerierung zusätzlich in den Bindevorgang vor der Bibliothek PROF3.LIB einzufügen. Beim RMOS3-Profiler als nachladbares Programm ist die Batch-Datei PROFLOAD.BAT aufzurufen. Normalerweise sind keine Änderungen durchzuführen.

#### 5.2.1 Einstellung der Systemparameter

Folgende Parameter können eingestellt werden (voreingestellte Werte in Klammern):

TASK\_NUM:           Anzahl der maximal möglichen Tasks im System (150)  
SVC\_GATE\_NUM:       Gate-Nummer für die SVCs (126)  
POOL\_ID:            Von diesem Speicherpool werden alle benötigten Speichersegmente angefordert(-1)  
MPICBASE:           Interrupt-Vektornummer für den Master-PIC (0x60)  
SPICBASE:           Interrupt-Vektornummer für den Slave-PIC (0x70)  
TIMERINT:           Interrupt-Nummer für den Betriebssystemtakt von RMOS (MPICBASE)

In der Tabelle xd\_pit werden zu dem Betriebssystemtakt in ms die zugehörigen Counterwerte des Timers angegeben. Der RMOS3-Profiler ermittelt automatisch den eingestellten Betriebssystemtakt in ms und entnimmt aus der Tabelle xd\_pit den zugehörigen Counterwert für die Umrechnung in ms.

#### 5.2.2 Anpassungen an andere Timer-Bausteine

Das Auslesen des Timer-Bausteins ist hardware-abhängig. Die Anpassung an andere Timer-Bausteine ist in der Datei PROFCFG.C möglich. Als Timer-Baustein wird der Typ 8254 (8253) unterstützt und mit #define ausgewählt.

Der Timer-Baustein 8254 entspricht dem Timer-Baustein im PC und ist defaultmäßig eingestellt. Der Counter-Wert eines anderen Timer-Bausteins kann nur verwendet werden, wenn der Timer-Baustein während des Betriebs ausgelesen werden kann (z.B. beim Timer-Baustein 8254 und beim im 80186 integrierten Timer möglich, ebenso bei der PC-HW).

### 5.3 Speicherbedarf

Der Speicherbedarf für die Task *RPROF* beträgt ca. 27 kByte Code (ohne C-Laufzeitbedarf), 1 kByte RAM und 2 kByte Stack.

Für den Speicherbedarf von *RPROF* zum Abspeichern der Taskaktivitäten (1 - 64 kByte), zum Abspeichern der Rechenzeiten pro Task (max. Taskanzahl \* 9 in Bytes) bzw. zum Abspeichern der Daten bei einer zyklischen Messung (max. Taskanzahl \* 9 \* Anzahl der Messungen in Bytes) ist ein Speicherpool entsprechend zu dimensionieren.

Die ID des Speicherpools ist defaultmäßig die Heap-ID (-1). Änderungen der Speicherpool-ID sind in der Datei PROFCFG.C möglich.

### 5.4 Der RMOS3-Profiler als nachladbare Task

Das RMOS-System muß in den RAM-Speicher geladen sein, d.h. es darf nicht aus dem EPROM abgearbeitet werden. In das RMOS-System dürfen keine Hooks (RM3HOOKS.LIB) installiert sein. Ansonsten sind keine Besonderheiten zu beachten.

Projekt/Titel:	RMOS3-Profiler	Filename:	RM3_PROFILER.DOC
Bezeichnung:	Handbuch	Version:	vorläufig
			Seite 22 von 26

---

## 5.5 Der RMOS3-Profiler als Task bei der Systemgenerierung

Der RMOS3-Profiler kann auch unter Verwendung von Hooks implementiert werden. Die Hooks und der RMOS3-Profiler als Task werden in das System eingebunden. Bei der RMOS-Version V3.0 können in diesem Fall keine Betriebszustände im S-Zustand aufgezeichnet werden (S-STATE, S-STATESW).

### 5.5.1 Konfigurierung des RMOS3-Profilers als Task (Aufruf x\_prof\_init)

Die Task *RPROF* wird mit dem Aufruf `x_prof_init(priority)` in der Initialisierungstask (INITTASK.C) initialisiert. Der Parameter *priority* gibt die Priorität an, mit der die Task *RPROF* erzeugt wird. Die Task katalogisiert sich mit "RProfStrt".

Zur Ermittlung der IDLE-Zeit ist die Installierung einer Busy-Task mit der niedrigsten Priorität (Priorität 1) im System notwendig. Dies geschieht automatisch beim Aufruf `x_prof_init(priority)`, falls die Busy-Task noch nicht in der Initialisierungstask erzeugt wurde (Aufruf `X_BU_INIT`).

Die im Lieferumfang enthaltene Initialisierungstask entspricht der Datei *RMCONF.C* (RMOS3 V3.20) bzw. *INITTASK.C* (RMOS3 V3.0), erweitert um den Aufruf `x_prof_init()`.

### 5.5.2 Konfigurierung der Hooks

In der RMOS3-Builderdatei für *RM3LPC1.BLD* sind die Hooks als Gates zu definieren (siehe *HOOKS.TXT* im RMOS-Produkt).

Weiterhin ist die Hooks-Bibliothek *RM3HOOKS.LIB* vor die *RM3BAS.LIB* zu binden.

Die im Lieferumfang enthaltene Datei *RM3LPC2.BLD* entspricht der Datei *RM3LPC1.BLD*, angepaßt für die Hooks.

---

## 6. Lieferumfang und Installation

Im Lieferumfang des RMOS3-Profilers sind enthalten:

1 Diskette 3 1/2"  
1 Handbuch in Deutsch

Auf der Diskette sind folgende Dateien enthalten:

INSTALL.BAT:	Batch-Datei zum Installieren des RMOS3-Profilers unter RMOS
RPROF.386:	Nachladbares Programm <i>RPROF</i>
PROF3.LIB:	Bibliothek mit RMOS3-Profiler (Task und Hooks) für RMOS3
PROFCFG.C:	C-Source für die Anpassung an andere Timer-Bausteine
PROF.H:	Headerdatei des RMOS3-Profilers
PROF.SUB:	Subsystem Definition des RMOS3-Profilers
PROFL.BAT:	Batchdatei zum Übersetzen von PROFCFG.C und Erzeugen eines nachladbaren Programmes RPROF.386
TESTMBX.BAT:	Batchdatei zum Erzeugen des Demos für die Mailboxschnittstelle
TESTMBX.C:	C-Source des Demos für die Mailboxschnittstelle
für RMOS3 V3.0:	
INITPROF.C:	Initialisierungstask mit Aufrufbeispiel für den RMOS3-Profiler
GEN_PC2.BAT:	Batchdatei für Systemgenerierung (angepaßt für den RMOS3-Profiler)
RM3LPC2.BLD:	Builderdatei (mit Hooks-Anpassung)
für RMOS3 V3.20:	
RMPROF.C:	Initialisierungstask mit Aufrufbeispiel für den RMOS3-Profiler
GENSYSI.BAT:	Batchdatei für Systemgenerierung (CADUL, angepaßt für den RMOS3-Profiler)
GENSYSC.BAT:	Batchdatei für Systemgenerierung (Intel, angepaßt für den RMOS3-Profiler)
RM3LPC.BLD:	Builderdatei (mit Hooks-Anpassung)

## Installation

Zur Installation sind folgende Arbeiten notwendig:

- Diskette in Laufwerk einlegen
- in das Root-Directory von RMOS wechseln
- Batch-Datei INSTALL.BAT vom Laufwerk mit der Diskette starten (z.B. A:INSTALL A:)

Im RMOS-Baum wird das Directory RPROF angelegt.



---

## 7. Fehlermeldungen

Wenn es beim Aufruf eines von der Task *RPROF* benutzten SVCs zu einem Fehler kommt, wird eine Fehlermeldung ausgegeben (z.B.: \*\*\* SVC error with pause !).

Falls bei der Rechenzeitermittlung *TASK\_NUM* zu klein ist, erscheint die Fehlermeldung "\*\*\*\* Not enough memory, increase TASK\_NUM". Die Anzahl der maximalen Tasks (*TASK\_NUM*) wird in der Datei *PROFCFG.C* konfiguriert.

Falls der Speicherpool zum Anfordern des Ausgabebuffers zu klein ist, erscheint die Fehlermeldung "\*\*\*\* Not enough memory !".

Falls eine Messung ausgegeben werden soll, ohne vorher eine Messung zu starten, erscheint die Fehlermeldung "\*\*\*\* Measurement not started !"

Falls eine Messung gestartet werden soll, obwohl noch eine Messung läuft, erscheint die Fehlermeldung "\*\*\*\* Measurement not stopped !"

Falls eine zyklische Messung ausgegeben werden soll, ohne vorher eine zyklische Messung zu starten, erscheint die Fehlermeldung "\*\*\*\* Cyclic measurement not started !"

Bei einer fehlerhaften Eingabe in den Menüs erscheint die Fehlermeldung "\*\*\*\* Wrong input !"

Falls eine Messung für zu kurze Zeit gestartet wurde, erscheint die Meldung "\*\*\*\* Duration not enough !"

Falls bei der Ermittlung des Intervalls für den Timertakt dieser nicht in der Tabelle *xd\_pit* in der Datei *PROFCFG.C* eingetragen ist, erscheint die Meldung "\*\*\*\* This timer tick not supported, change table xd\_pit (file profcfg.c) !"

Bei einer unbekannten RMOS-Version erscheint die Fehlermeldung "\*\*\*\* Unknown RMOS Version !"

Wenn bei der Konfigurierung des RMOS3-Profilers als Task die Hooks nicht konfiguriert wurden, erscheint die Fehlermeldung "\*\*\*\* Hooks not activated !"

Wenn beim Menüpunkt "Output measurement to file *RPROF.SAV*" kein aktueller Pfad (Current directory) existiert, erscheint die Fehlermeldung "\*\*\*\* File error !".

---

## 8. Abkürzungsverzeichnis

CLI	Command Line Interpreter
ICE	In Circuit Emulator
SRM	System Memory Resources
SVC	SuperVisor Call (Betriebssystemaufruf)