Die verwaltete Version des RED-VMK

Dieses Dokument beschreibt **die verwaltete Version des Redundancy Virtual Machine Kernel**, abgekürzt **RED-VMK**.

Inhalt

© GEFAZ mbH, 91301 Forchheim, Äußere Nürnberger Str. 62,
email: raimund.glenz@gefaz.de, Tel: +49 (0)9191/3539860

Einführung

Der **RED-VMK** ist eine intelligente Softwarelösung zur Realisierung von hochverfügbaren Anwendungen im Industriebereich. Er basiert auf dem Betriebssystem Windows und ist für die Automatisierungstechnik und den Maschinenbau einsetzbar.

Bei vielen Anwendungen kommen in der Automatisierungstechnik Steuerungssysteme zum Einsatz, die es ermöglichen, die Applikation fortzuführen, wenn Komponenten der Steuerung ausfallen. Ausfälle entstehen durch Versagen von Hardware Komponenten und/oder werden durch Fehler in der Steuerungsapplikation verursacht.

Der Aufbau einer hochverfügbaren Automatisierungstechnik Lösung besteht im Wesentlichen aus einer Master/Slave Konfiguration, wobei der Master die Steuerungsapplikation ausführt und der Slave sich im Standby Betrieb befindet. Bei Auftreten eines Fehlers im Master System wird eine Redundanzumschaltung auf das Slave System durchgeführt, wobei der Slave die Applikation fortführt.

Eine intelligente Lösung soll neben dem Überwachen von Hardware Komponenten auch das Laufzeitverhalten der Steuerungsapplikation überwachen, so dass eine Redundanzumschaltung und eine Fortführung der Applikation bei Hardware Fehlern und bei Fehlern in der Steuerungsapplikation ermöglicht wird.

Der **RED-VMK** erfüllt diese Anforderungen durch Bereitstellen einer Laufzeitumgebung, in der die Applikation des Anwenders ausgeführt wird.

Der **RED-VMK** überwacht unter anderem relevante Kenngrößen der Anwenderapplikation wie z.B. benötigte CPU Zeit, Ressourcenbedarf und allgemeine Schutzverletzungen, die durch Division durch 0 oder Überschreiben von Datenbereichen oder durch andere Ursachen ausgelöst werden.

Der Anwender hat die Möglichkeit, seine Applikation in der Programmiersprache C zu implementieren. Dadurch ist es möglich, komplexe Applikationen mit Hilfe von modernen Software Engineering Methoden und auf der Basis von wieder verwendbaren Komponenten zu realisieren.

Abkürzungen

RED-VMK Redundancy Virtual Machine Kernel

Dokumente

[1] Hochverfügbarkeit, Fehlertoleranz und Safety

[2] Die Verwaltete Version des RED-VMK

[3] Die Interpreter Version des RED-VMK

[4] Anwendungsgebiete des RED-VMK

[5] Das RED-VMK Applikations Interface

[6] Das Applikationsbeispiel „Lauflicht“

© GEFAZ mbH, 91301 Forchheim, Äußere Nürnberger Str. 62,
email: raimund.glenz@gefaz.de, Tel: +49 (0)9191/3539860

RED-VMK Varianten

Der RED-VMK arbeitet in den zwei Varianten:

- verwalteter (native) Modus
- interpretierter Modus

Im **interpretierten** Modus werden die Anweisungen der Anwender Applikation nicht von der CPU des PCs ausgeführt, sondern von einem Interpreter des **RED-VMK**. Dadurch ist die volle Kontrolle des Laufzeitsystems des **RED-VMK** über die Anwender Applikation gewährleistet.

Die interpretierende Variante ist für Anwendungen geeignet bei der Umschaltzeiten von bis zu **1ms** und Echtzeitverhalten gefordert ist.

Im **verwalteten** Modus werden die Anweisungen der Anwender Applikation von der CPU der Plattform ausgeführt und die Prozessdaten der Applikation vom Laufzeitsystem des **RED-VMK** verwaltet. Dadurch ist zwar keine volle Kontrolle über die Anwender Applikation möglich, aber es wird die Datenkonsistenz zwischen der Master Applikation und der Slave Applikation bei einer Redundanz Umschaltung gewährleistet. Der verwaltete Modus wird auch als **native** Modus bezeichnet

Die verwaltende Variante ist für Anwendungen geeignet bei der Umschaltzeiten von bis zu 100 ms keine Probleme für die Anwendung ergeben.

Gegenstand dieses Dokumentes ist der verwaltete Modus.

Der interpretierte Modus ist im Dokument [3] beschrieben.

Systemplattform

Der **RED-VMK** wird als eigene Laufzeitumgebung auf einer industriellen Windows Plattform ausgeführt, wobei Windows 7 und nachfolgende Windows Versionen unterstützt werden. Der **RED-VMK** besteht aus folgenden Komponenten:

Komponente	***Applikationstyp***
RED-VMK	Windows Win32/Win64 Prozess
Supervisor	Windows Kernel Mode Treiber
Redundanz Manager	Windows Win32/Win64 Prozess

Abbildung 1: RED-VMK Komponenten

Der **RED-VMK** wird als Windows Prozess (User Mode Applikation) ausgeführt und stellt die Laufzeitumgebung für die Anwender Applikation zur Verfügung. In dieser Laufzeitumgebung wird die Applikation als Windows DLL ausgeführt.

Der **Supervisor** besteht aus einem Windows Kernel Mode Treiber, der die Prozessdaten der Anwender Applikation verwaltet. Da Windows Kernel Mode Treiber im Allgemeinen im Adressraum des Windows Kernels ausgeführt werden, können Daten eines Kernel Mode Treibers nicht direkt von einer Windows User Mode Applikation geschrieben werden.

Das heißt, es ist physikalisch **nicht** möglich, dass die Anwender Applikation oder sogar der RED-VMK selbst durch ein Fehlverhalten die Prozessdaten, die im Supervisor verwaltet werden, unbeabsichtigt überschreibt und so inkonsistent werden lässt.

Der **Redundanz Manager** wird als Windows Prozess ausgeführt und kommuniziert durch das Redundanzprotokoll mit dem Slave System oder respektive mit dem Master System. Der Redundanz Manager überträgt die Prozessdaten zum korrespondierenden System, so dass im Fehlerfall die Prozessdaten im Slave System verfügbar sind.

Der Anwender kann ein redundantes **RED-VMK** Steuerungssystem mit handelsüblichen PCs oder Laptops aufbauen. Als Zielsystem wird dem Anwender jedoch dringend empfohlen, industrielle PC Systeme einzusetzen. Diese Systeme zeichnen sich durch höhere Hardwarequalität und Lebensdauer aus.

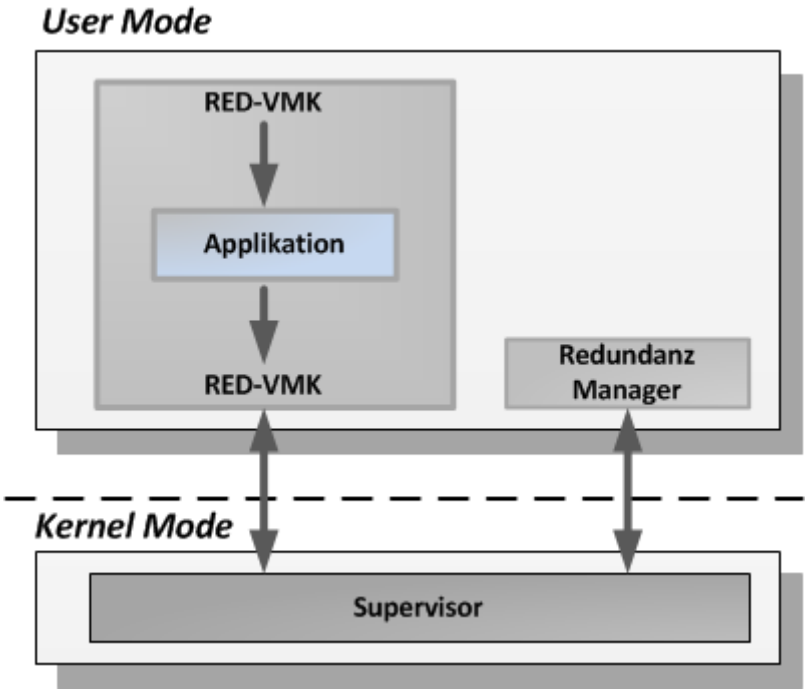

Abbildung 2, Systemkomponenten des RED-VMK

System Architekturen des RED-VMK

Eine für den **RED-**VMK implementierte Steuerungsapplikation kann entweder als „**Single Mode**“ Konfiguration **ohne** Redundanzfunktionalität oder als „**zweikanalige**“ Konfiguration **mit** Redundanzfunktionalität ausgeführt werden.

Die Single Mode Konfiguration

Das folgende Bild zeigt die **Single Mode** Architektur des RED-VMK.

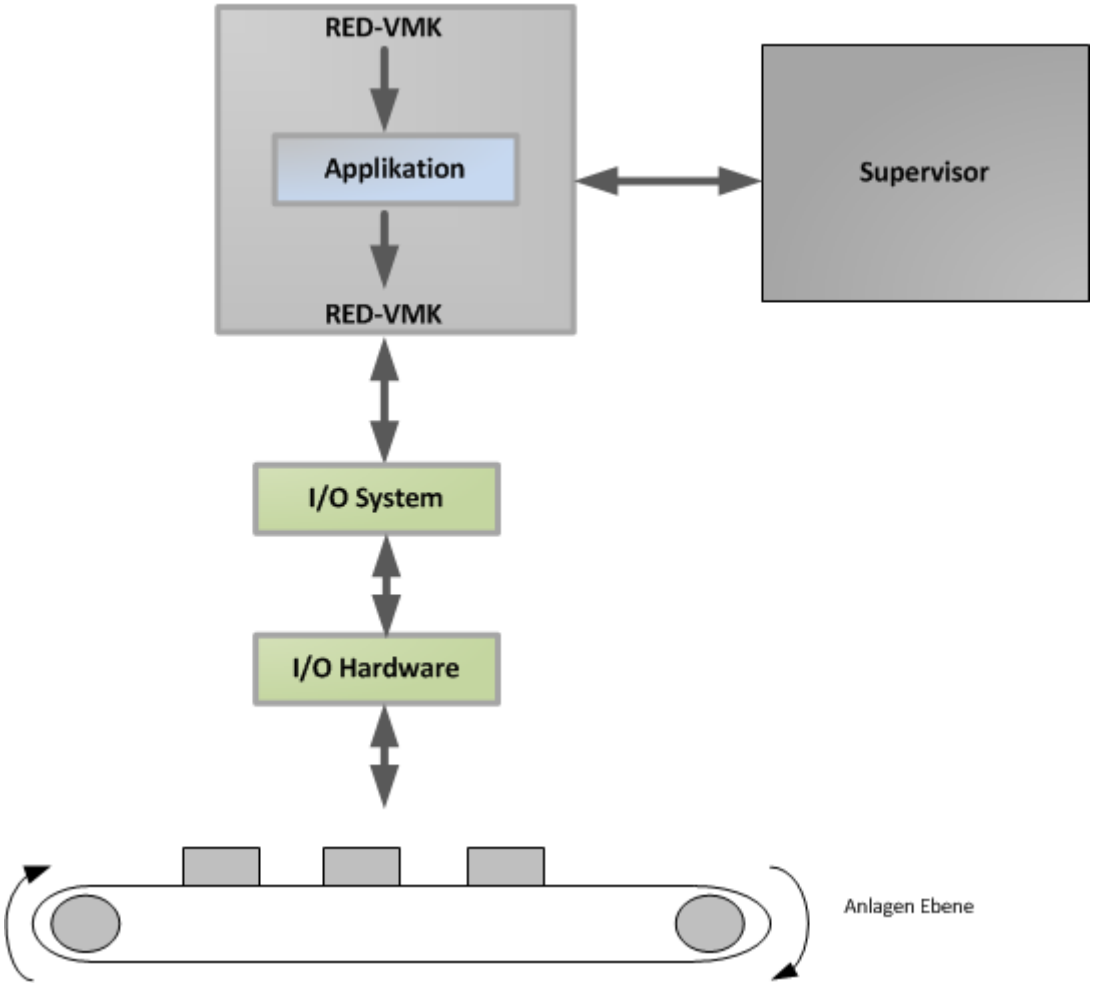

Abbildung 3, RED-VMK im Single Mode Betrieb

Die Anwender Applikation wird vom **RED-VMK** ausgeführt. Die Applikation greift über entsprechende Laufzeitfunktionen des **RED-VMK** auf die I/O Hardware zu, um Eingangssignale zu lesen und Ausgangssignale zu schreiben. Da die Applikation nicht im Redundanzbetrieb ausgeführt wird, wird bei einem Fehler z.B. in der I/O Hardware oder in

der Applikation die Steuerungsapplikation vom **RED-VMK** angehalten. Da das System im „stand alone“ Betrieb arbeitet, erfolgt keine Redundanzumschaltung.

Die Ausgangssignale werden dabei in einen vom Anwender definierbaren Zustand geschaltet, um den zu steuernden Prozess in einen sicheren Zustand zu überführen.

Die zweikanalige, redundante Konfiguration

Das folgende Bild zeigt die zweikanalige, redundante Architektur des **RED-VMK**.

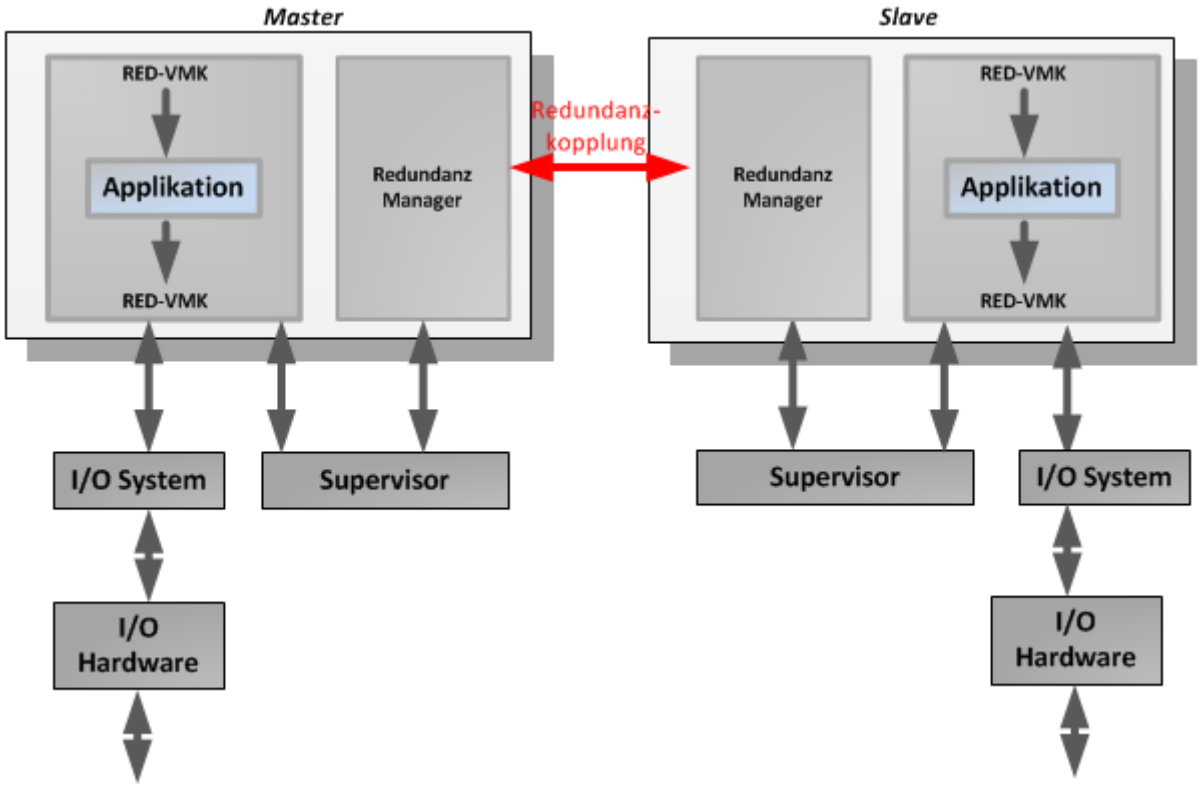

Abbildung 4, Redundante Anwenderapplikation

In der Redundanz Konfiguration besteht das System aus einem Master und einem Slave. Die Hardware Plattform des Masters und des Slave besteht je aus einem Windows PC System.

Der Master führt in dieser Konfiguration die Anwender Applikation aus, wobei Eingangssignale über das I/O System gelesen und Ausgangssignale geschrieben werden. Der **RED-VMK** überwacht dabei die Anwender Applikation und die I/O Hardware, der Supervisor überwacht den **RED-VMK** selbst. Der **RED-VMK** übermittelt die Prozessdaten der Anwender Applikation an den Supervisor.

Master und Slave sind durch eine sogenannte Redundanzkopplung miteinander verbunden. Der Redundanz Manager des Masters nimmt die Prozessdaten der Anwender Applikation vom Supervisor entgegen und überträgt diese mit Hilfe des Redundanz Protokolls an den Slave. Der Redundanz Manager des Slave übergibt die Prozessdaten der Anwender Applikation des Masters an den Supervisor des Slave.

Die Anwender Applikation des Slaves befindet sich dabei im Standby und wird nicht vom **RED-VMK** ausgeführt.

Erkennt der **RED-VMK** einen Fehler in der I/O Hardware oder in der Anwender Applikation, wird eine Redundanzumschaltung vom Master zum Slave durchgeführt. Dabei werden die letzten Prozessdaten zusammen mit einer Störmeldung an den Slave übergeben. Der Master hält dabei die Anwender Applikation an und der Slave führt die Anwender Applikation mit den letzten, aktuellen Prozessdaten fort.

Im Folgenden werden die Fehlerklassen, die der **RED-VMK** und seine Subsysteme erkennen kann und die entsprechenden Reaktionen beschrieben.

Fehler in der Anwender Applikation

Der RED-VMK überwacht folgende Kenngrößen der Applikation:

- Zykluszeit der Applikation
- Ressourcenverbrauch
- Schutzverletzungen, wie z.B. illegale Speicherzugriffe, Division durch 0 etc.
- Nicht mehr stattfindende Zugriffe auf I/O Signale

Tritt einer der genannten Fehler in der Anwender Applikation auf, informiert der **RED-VMK** den Supervisor über einen Ausfall der Applikation. Der Supervisor übergibt eine entsprechende Statusmeldung an den Redundanz Manager, der diese Information zum Redundanz Manager des Slave überträgt. Der Redundanz Manager des Slave informiert seinen eigenen Supervisor, der dem **RED-VMK** des Slave den Ausfall der Master Applikation signalisiert.

Der **RED-VMK** des Slave führt daraufhin die sich im Wartezustand befindende Slave Applikation fort. Der **RED-VMK** des Masters hält die Anwender Applikation an und bootet das System neu. Nach erfolgtem Hochlauf übernimmt der bisherige Master die Rolle des Slave und synchronisiert sich mit dem aktuellen Master, dem vorhergehenden Slave, neu.

Sekundärfehler im RED-VMK

Ein Sekundärfehler im **RED-VMK** liegt vor, wenn ein Absturz der Anwender Applikation die Reaktionsfähigkeit des **RED-VMK** blockiert. In einem Single Core System könnte die Anwender Applikation die CPU blockieren, so dass der **RED-VMK** zwar noch betriebsbereit ist, aber keine Rechenzeit mehr bekommt.

Da der Supervisor im Windows Kernel im Interrupt Betrieb arbeitet, wird diesem über den Interrupt Handler in jedem Fall CPU Zeit zugeteilt. Dadurch erkennt der Supervisor anstelle des **RED-VMK** einen System Ausfall und gibt diese Information wie bereits beschrieben über den Redundanz Manager an das Slave System weiter.

Fehler in der I/O Hardware

Das I/O System des RED-VMK ermöglicht Zugriffe auf I/O Signale. Das I/O System kommuniziert dabei mit der dazu gehörenden I/O Hardware. Das I/O System des **RED-VMK** überwacht die Kommunikation mit der I/O Hardware.

Treten beim Schreiben oder Lesen von I/O Signalen Fehler auf, die z.B. von der I/O Hardware gemeldet werden, wird vom **RED-VMK** eine entsprechende Status Meldung an den Supervisor übergeben, der wie bereits beschrieben diese Information über den Redundanz Manager an den Slave weiter leitet.

Ausfall der Redundanzkopplung

Der Redundanz Manager des Masters kommuniziert mit dem Redundanz Manager des Slave. Erkennt der Redundanz Manager des Slave, dass keine Kommunikation mehr mit dem Master stattfindet, gilt der Master als ausgefallen und die Anwender Applikation des Slave Systems wird fortgeführt.

Ein Ausfall der Kommunikation entsteht durch einen Ausfall des Redundanz Managers des Master oder durch eine Unterbrechung der Master/Slave Verbindung, wie z.B. das Ziehen des Netzwerk Kabels.

Spannungsausfall

Ein Spannungsausfall einer CPU in der Redundanzkonfiguration wird vom noch aktiven Partner erkannt. Dieser Fehler wird gemäß des Ausfalls der Redundanz Kopplung behandelt.

Applikationsentwicklung und Programmiermodell

Wie bereits beschrieben wird die Anwender Applikation in der Programmiersprache ANSI C erstellt. Der RED-VMK stellt für die Applikation entsprechende Bibliotheken zu Verfügung, die die Funktionalität des **RED-VMK** in Form von System Funktionen bereitstellen. Die Anwender Applikation sollte nur die vom **RED-VMK** zu Verfügung gestellten System Funktionen verwenden.

In der ersten Version des **RED-VMK** ist es grundsätzlich möglich, Funktion der C-Runtime Library oder WIN32 Funktionen (Windows System Funktionen) zu verwenden, jedoch kann in diesem Fall die Redundanz Funktionalität nicht vollständig gewährleistet werden. Es ist die Aufgabe des Anwenders dafür zu sorgen, dass im Redundanzfall das Applikationsprogramm sinnvoll weitergeführt werden kann.

Spätere Versionen stellen eigene Runtime Libraries zur Verfügung. Diese unterstützen auch den redundanten Betrieb.

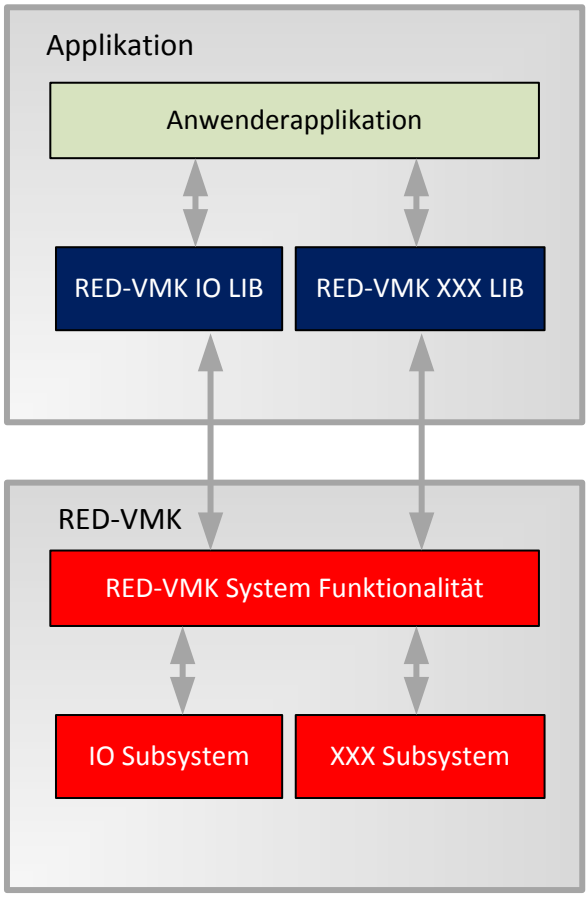

Abbildung 5, Applikationsbibliotheken

Die Anwender Applikation wird als Win32 DLL erstellt und vom **RED-VMK** geladen und ausgeführt und kann mit dem Visual Studio von Microsoft implementiert und mittels des Visual Studio Debuggers getestet werden.

Inbetriebnahme der Applikation

Für die Inbetriebnahme stellt der **RED-VMK** eine Projektierungssoftware zur Verfügung. Mit dieser Software werden I/O-Signale, I/O-Module und Feldbuskontroller projektiert. Zum Debuggen können Funktionen wie „Steuern“ und „Forcen“ von Signalen angewendet werden. Weiterhin können Traces aus dem Applikationsprogramm in der Projektierungssoftware angezeigt werden.

Selbstverständlich stehen auch die komfortablen Debugging Funktionen des Visual Studios wie z.B. Breakpoints, Watches zur Verfügung.