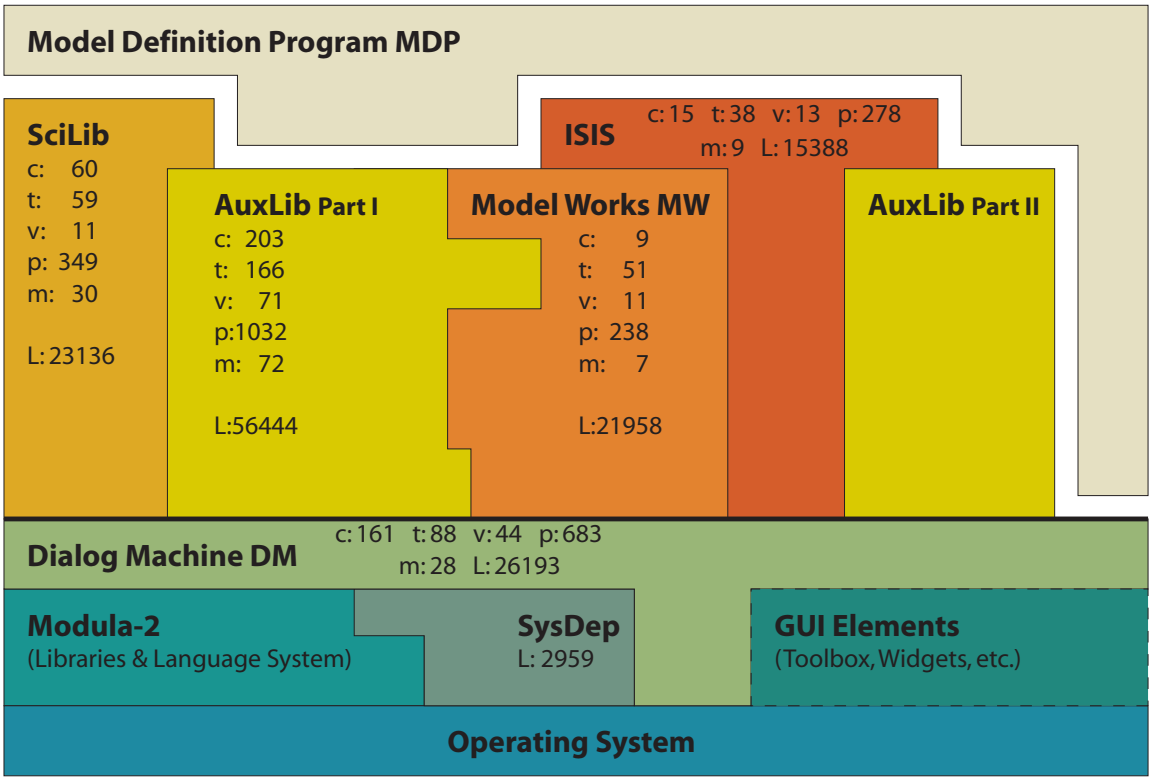
Technische Semesterarbeit UMNW ETHZ

Analyse und Evaluation der RAMSES Software bezüglich Neuimplementation auf modernen OS X und UNIX/Linux Arbeitsplatzsystemen

Oliver Gardi
ogardi@student.ethz.ch

SS / WS 2004



Gruppe Systemökologie des Instituts für terrestrische Ökologie
unter der Leitung von Dr. Andreas Fischlin

Inhaltsverzeichnis

1 Einleitung

Während den letzten 20 Jahren wurde im Fachbereich Systemökologie des Institutes für Terrestrische Ökologie ITÖ der ETH Zürich unter der Leitung von Dr. Andreas Fischlin eine Plattform unabhängige Modellierungs- und Simulationsumgebung für Arbeitsplatzrechner entwickelt. Die mit dem Akronym RAMSES[1] bezeichnete Software erlaubt Systeme von nicht linearen Differentialgleichungen DESS[2], Differenzengleichungen SQM[3] und diskreten Ereignissystemen DEVS[4] interaktiv zu lösen (Fischlin et al., 1994, auch 1991).

Die Simulationsmodelle und RAMSES selbst sind in der an der ETH Zürich entwickelten Sprache Modula-2 implementiert (Wirth, 1983; Wirth et al., 1992). Modula-2 ist eine hochentwickelte und mächtige Programmiersprache welche mit Hilfe von Modulen die Entwicklung von strukturiertem prozeduralem Code ermöglicht. Trotzdem wird die Sprache heutzutage nur noch selten gebraucht und die Auswahl an Kompilern ist sehr beschränkt. Insbesondere gibt es zur Zeit keinen Modula-2 Kompiler, welcher die Zielplattformen MacOS X/PPC, Solaris/SPARC, Windows/x86 und Linux/x86 allesamt unterstützt.

Die momentane Hauptplattform von RAMSES ist der Macintosh/68k, auf welchem mittels dem Kompiler MacMETH Code für 68k Motorola Prozessoren generiert wird. Zwar wird dieser Code zur Zeit auch noch von neueren Plattformen wie z.B. der Classic-Umgebung des MacOS X unterstützt, kann aber von gewissen Eigenschaften moderner Plattformen nicht profitieren. Weitaus schlimmer ist jedoch die Tatsachte, dass mit der Portierung vieler Programme von der Classic auf die MacOS X Plattform der Classic-Emulator unter MacOS X an Bedeutung verliert.[5]

Weitere von der RAMSES-Software unterstützte Plattformen sind Solaris/SPARC mit dem EPC Kompiler und Windows/x86 mit dem Stony Brook Kompiler. Auch bezüglich diesen Plattformen müssen mittelfristig neue Lösungen gesucht werden. Der EPC Kompiler wird mittlerweile nicht mehr gewartet, keine neuen Lizenzen können gekauft werden und der Quelltext liegt nicht frei.

Ziel dieser Semesterarbeit ist es, Grundlagen für eine anschliessende Portierung von RAMSES auf die oben genannten Zielplattformen zu erarbeiten. Dies umfasst die Analyse des aktuellen Bestandes an Quelltext sowie die Evaluation verfügbarer Kompiler. Am Schluss sollen ein bis zwei Kompiler am Quelltext getestet werden. Auch die ersten Schritte in Richtung Portierung der RAMSES-Software auf MacOS X sollen unternommen werden um den Aufwand einer kompletten Portierung von RAMSES und die Qualität des vom Kompiler generierten Codes abschätzen zu können.

[1] **RAMSES: R**esearch **A**ids for **M**odeling and **S**imulation of **E**nvironmental **S**ystems
[2] **DESS: D**ifferential **E**quation **S**ystem **S**pecification
[3] **SQM: Seq**uential **M**achine
[4] **DEVS: D**iscrete **Ev**ent **S**ystem Specification
[5] So können z.B. die G5- und die G4-Rechner der neuen Generation nicht mehr direkt mit Classic (MacOS 9) gebootet werden, sondern unterstützen Classic nur noch über den Emulator innerhalb des MacOS X Betriebssystems.

2 Analyse des Ist-Zustandes

Die Analyse des Ist-Zustandes verfolgt verschiedene Ziele. Einerseits soll sie einen qualitativen Überblick über die bestehende Software-Architektur und das Zusammespiel der einzelnen Softwareschichten verschaffen und somit die Funktionsweise der RAMSES-Software verdeutlichen. Andererseits soll eine quantitative Erhebung von Kennzahlen Aufschluss über die Mächtigkeit der einzelnen Softwareschichten und den im Falle einer Portierung zu erwartenden Aufwand geben.

Grundsätzlich ist der Zustand der Software, sowohl die Architektur wie auch die Implementation und deren Umfang, von zentraler Bedeutung für die Evaluation von verschiedenen Portierungsoptionen, da der mit der Portierung verbundene Arbeitsaufwand in hohem Masse von der jetzigen Struktur der Software und deren Abhängigkeit von spezifischen Plattformen[6] abhängt.

2.1 Die RAMSES Architektur

Die Abbildung 1 zeigt den schematischen Aufbau der RAMSES Softwarepakete. Deutlich zu erkennen ist die hierarchische Strukturierung der Gesamtsoftware in verschiedene in sich abgeschlossene Softwareschichten, welche über klar definierte und möglichst knappe Schnittstellen miteinander kommunizieren. In der Abbildung 1 klar zum Ausdruck kommt die Abschirmung der RAMSES-Software von der Plattform (d.h. vom Betriebsystem, der Systemsoftware und von Bibliotheken des Kompilers) durch die Dialog Machine. Dies ist die, für die Portabilität und den Wartungsaufwand verschiedener Ports zentralste architektonische Eigenschaft der RAMSES-Software.

Dialog Machine (DM)

Die "Dialog Machine" DM ist von zentraler Bedeutung für das Funktionieren der RAMSES Software. Sie bildet das Fundament der hierarchischen RAMSES Software-Architektur und ist für die *Low-Level Kommunikation* mit dem Betriebssystem, dem Kompiler und anderen, RAMSES externen Bibliotheken verantwortlich. Ihr unterliegen u.a. die Speicherverwaltung und die System Interrupts. Insbesondere ist sie auch für sämtliche I/O-Funktionen, inkl. der von RAMSES erzeugten grafischen Ausgabe verantwortlich. Die Vielfalt von Funktionen und Prozeduren, welche die plattformspezifischen Bibliotheken zur Verfügung stellen, werden von DM zu einer schlanken und abstrakten Schnittstelle reduziert. Durch diese Schnittstelle kann die direkte Interaktion von höheren Softwareschichten mit der Plattform weitgehend vermieden werden.

Im Gegensatz zur DM Version auf der Mac und der PC Plattform, implementiert die UNIX Version der DM, die sogenannte *BatchDM*, keine grafische Benutzerschnittstelle. Das Vorhandensein von Vorgabewerten wird vorausgesetzt, Benutzerinteraktionen während den Simulationsläufen sind nicht zugelassen. Die BatchDM dient der Stapelverarbeitung, z.B. von Modellen mit RASS[7]. Die von der BatchDM veröffentlichte Schnittstelle ist jedoch mit jener der DM auf der Mac- und PC-Plattform identisch. Einzig und alleine die Implementation der Module unterscheidet sich an gewissen Stellen.

Auch die *interne Architektur* der DM folgt einer gewissen Hierarchie. So sind die meisten plattformabhängigen Funktionen in das Subpackage SysDep ausgelagert, was wesentlich zur Portabilität der Software beiträgt. Entgegen den eigentlichen Design-Prinzipien kommt es aus Effizienzgrüden auch vor, dass DM Module ausserhalb von SysDep direkt auf Funktionen des Betriebssystems zugreifen. Daneben gibt es weitere plattformabhängige Module, welche auf der Mac Plattform die Kommunikation mit der

[6] Unter Plattform wird im Folgenden jeweils die gesamte Umgebung einer RAMSES Implementation verstanden, d.h. Rechnerarchitektur (z.B. x86), Betriebsystem inkl. Bibliotheken (z.B. Windows) und Kompiler inkl. Bibliotheken (z.B. Stony Brook)

[7] **RASS: RA**MSES **S**imulation **S**erver

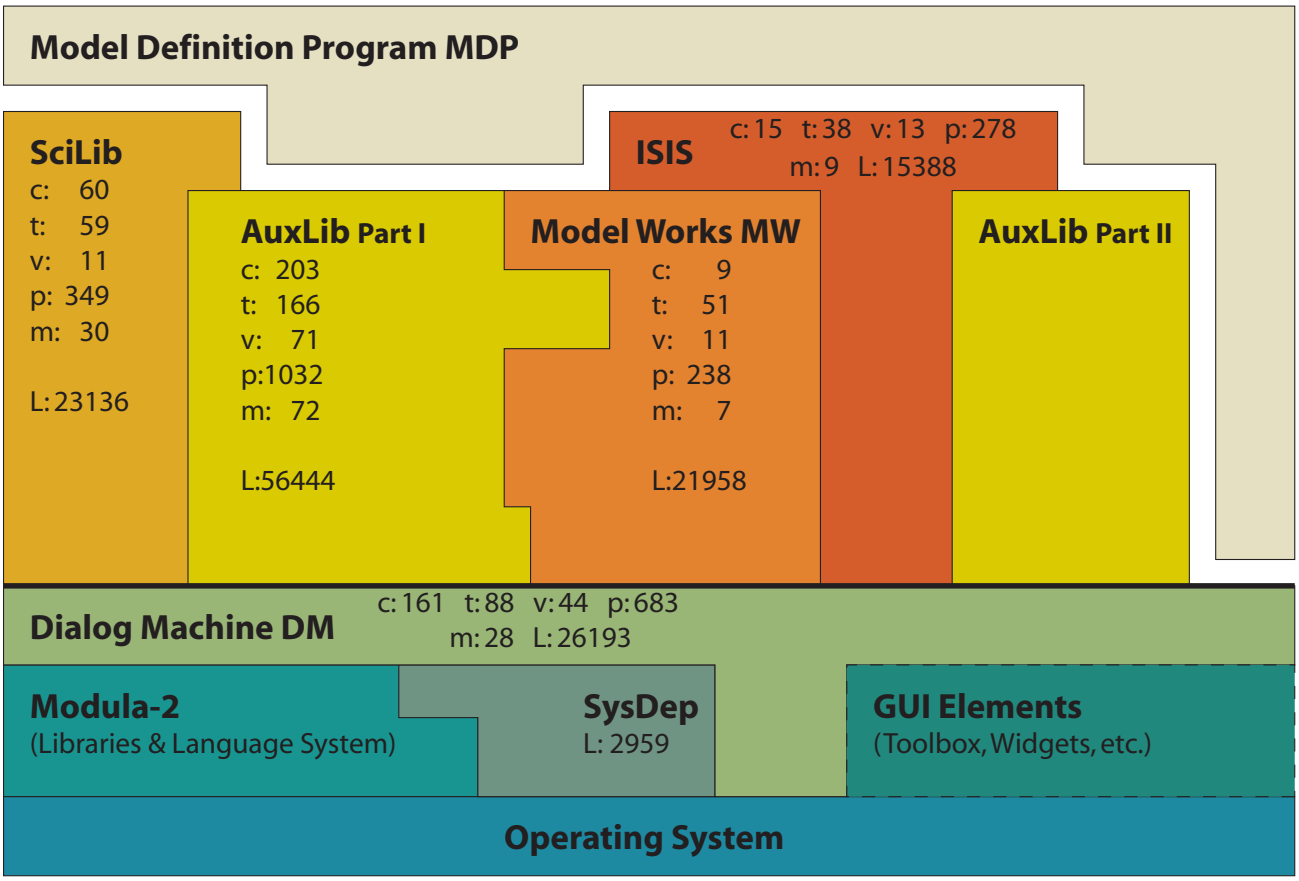


Abbildung 1: RAMSES Software-Architektur. Diese Grafik zeigt den Grad der Abhängigkeit (horizontale Kontaktflächen) der verschiedenen RAMSES Software-Schichten untereinander. Die Flächen der einzelnen Schichten ist in etwa proportional zum Umfang der Implementation, d.h. der Anzahl Zeilen Quelltext. Die Kennzahlen zeigen einerseits den funktionellen Umfang der angebotenen Schnittstelle: Konstanten (c), Typen (t), Variablen (v) und Prozeduren (p), die Anzahl Module durch welche diese Funktionalität angeboten wird (m), sowie die Zeilen Quelltext, die für die Implementation benötigt wurden (L).

Toolbox übernehmen. Dies sind unter anderem DMQuickDraw für die Grafik, DMMemTypes für das Speicher-Management, DMHFSBase und DMBase für die Verwaltung von Dateien. Auch die Kommunikation mit dem Kompiler, in erster Linie Dateisystem- und Menu-Operationen, ist in bestimmten Modulen gebündelt.

Die Auslagerung von plattformspezifischen Abhängigkeiten der gesamten RAMSES-Software in die Dialog Machine, sowie die Konzentration dieser Abhängigeiten in gewissen Modulen innerhalb der DM, führt zu einen grossen Grad an *Plattformunabhängigkeit (Portabilität)* der gesamten RAMSES-Software und minimiert den bei einer Portierung zu erwartenden Aufwand im Idealfall auf gewisse Anpassungen in der DM. Diese Kapselung der Platformabhängigkeit ist mit ein Grund, dass die RAMSES-Software mit einem Minimum an Aufwand auf 3 verschiedenen Plattformen (Mac, PC, Sun) gewartet werden kann.

ModelWorks (MW)

ModelWorks MW ist eine auf der DM basierende Modellierungs- und Simulationsumgebung (Fischlin et al., 1994). Sie erlaubt einem Endbenutzer die Definition von Modellen in Form eines Model Definition Programms MDP und führt anschliessend die Simulation dieser Modelle durch. Für eine ausführliche Beschreibung der ModelWorks-Umgebung sei auf die Originalliteratur verwiesen.

Als höhere Softwareschicht macht MW in erster Linie von Schnittstellen der DM gebrauch. Direkte Zugriffe auf Bibliotheken der Plattform sollten keine vorkommen. Nebst den Funktionen der DM benötigt MW lediglich ein Modul aus der AuxLib Bibliothek (Matrices).

Integrative Systems Implementation Software (ISIS)

ISIS ist eine weiterentwickelte objektorientierte Simulationsumgebung, welche auf der ModelWorks-Umgebung aufbaut und diese um zahlreiche Funktionen erweitert. ISIS ist allgemeiner geschrieben und systemtheoretisch besser veranktert als die MW-Umgebung und weist eine klare hierarchische Gliederung auf, welche das Verschachteln von Modellen erlaubt.

AuxLib und SciLib

Obwohl sehr umfangreich sind die Softwareschichten AuxLib und ScienceLib nicht als Kernbestandteile der Simulationsumgebung RAMSES zu verstehen. Viel mehr stellen diese Packete dem Benutzer Werkzeuge und Strukturen in Form von Objekten (Konstanten, Typen, Variablen, Prozeduren) zur Verfügung, welche den Umgang mit den eigentlichen Simulationspacketen ISIS und MW erleichtern. Wie der Name schon sagt, exportiert die ScienceLib Funktionalitäten, welche für die Implementation und Berechnung wissenschaftlicher Modelle benötigt werden. Dazu gehören z.B. auch Funktionen für die statistische Auswertung von Simulationsergebnissen. Die AuxLib ist in ihrem Aufbau sehr heterogen und stellt eine Menge Bezeichner zur Verfügung, die keinen funktionellen Zusammenhang aufweisen. Wie zentral die AuxLib trotzdem ist für das Gesamtpacket zeigt sich daran, dass mit Ausnahme der Dialog Machine alle Softwareschichten auf die Funktionalität der AuxLib angewiesen sind.

Die AuxLib setzt sich zusammen aus der AuxLibSE (Systems Ecology AuxLib) und der AuxLibPD (Public Domain AuxLib), von der jedoch kaum Gebrauch gemacht wird. Die AuxLib ist nebst der Funktionalität der DM auch auf Schnittstellen der ModelWorks-Schicht angewiesen. Die beiden Pakete stehen also in gegenseitiger Abhängigkeit. Die SciLib hingegen ist lediglich von der DM und der AuxLib abhängig.

2.2 Kenngrössen des Quelltextes

Bei der quantitativen Beurteilung des Quelltextes sind zwei Aspekte von Bedeutung: der Umfang der gegen aussen veröffentlichten Schnittstellen (Mass für die Funktionalität einer Softwareschicht) und die dazugehörige Implementation (Aufwand, mit welchem die Funktionalität zur Verfügung gestellt wird).

Gesamtumfang der Softwareschichten

Für jedes Paket der Dialog Machine und für verschiedene Subpakete von speziellem Interesse wurden mit Hilfe eines UNIX Shell Scripts (Anhang A.2) Kenngrössen folgender Typen von Modulen ermittelt (Tabelle 1).

Definitions-Module (.DEF) enthalten die Definition der Schnittstellen, welche von den Modulen importiert werden können. Die Schnittstellen beinhalten im wesentlichen Deklarationen von Prozeduren,

Variablen, Konstanten und Typen. Durch diese Bezeichner wird die Funktionalität eines Moduls definiert. Als Kennzahl für die Funktionalität interessiert deshalb nebst der Anzahl Zeilen des Quelltextes die Anzahl, der von den Definitionsmodulen exportierten Bezeichner.

Implementations-Module (.MOD) implementieren i.d.R. eine Schnittstelle. Ein Definitions-Modul kann von mehreren Implementations-Modulen implementiert werden. Dabei werden die im Definitions-Modul deklarierten Prozeduren mit den für die Funktionalität benötigten Algorithmen versehen. Bei der Implementation der Module interessiert in erster Line der Aufwand, mit welchem die Funktionalität eines Moduls, charakterisiert durch die exportierten Bezeichner im dazugehörigen Definitions-Modul, realisiert wurde, d.h. die Anzahl Zeilen Quelltext.

Objekt-Module (.OBM) enthalten den vom Kompiler erzeugten Binärcode, dessen Grösse für die Ausführung des Programms durch den Anwender von Interesse ist. Zwischen .MOD und .OBM Modulen besteht nicht zwingend eine 1:1 Beziehung, da mehrere .MOD Module in ein und dasselbe .OBM Modul übersetzt werden können.

Exportierte Schnittstellen

Von besonderem Interesse ist der Umfang der Schnittstellen (Definitions-Module), die von den einzelnen Softwareschichten exportiert werden. Über diese Schnittstellen kommunizieren die verschiedenen Softwareschichten untereinander.

Viele der in den Definitions-Modulen (.DEF) definierten Prozeduren, Typen, Konstanten und Variablen sind nur für den Gebrauch innerhalb der jeweiligen Softwareschicht vorgesehen (private Module). Diese Schnittstellen werden in der DM auch als Base.DEFs bezeichnet, während die von aussen sichtbaren Definitionen (öffentliche Module) als High.DEFs bezeichnet werden. Diese Nomenklatur wird jedoch in der RAMSES-Software nicht durchgehend angewendet, weshalb für die in der Tabelle 2 aufgeführte Analyse der öffentlichen Schnittstellen die QuickReferences, d.h. die im RAMSES Release publizierten Schnittstellen der jeweiligen Softwareschichten verwendet wurden.

Ein Vergleich der in der DM-QuickReference aufgeführten Bezeichner (Tabelle 2) mit jenen der DM High.DEF-Module (Tabelle 1) zeigt, dass die QuickReferences nicht eins zu eins mit den eigentlichen Schnittstellen übereinstimmen, die Grössenordnung jedoch gut wiederspiegeln.[8]

Die Tabelle 2 gibt Auskunft über die Anzahl der Module und deren Objekte, die von den verschiedenen Softwareschichten gegen Aussen zur Verfügung gestellt werden. Auffällig ist der grosse Funktionsumfang der Basis-Schichten DM und AuxLib im Vergleich zu den eigentlichen Simulationsumgebungen MW und ISIS. Dahinter versteckt sich eine mit der Komplexität der Schichten zunehmende Abstraktion der Schnittstelle. DM und AuxLib stellen eine grosse Menge elementarer Funktionen für die höheren Softwareschichten zur Verfügung. Mit Hilfe einer Vielzahl solcher Elementar-Funktionen wird von den höheren Softwareschichten eine beschränkte Anzahl abstrakter, in ihrer Funktionalität jedoch sehr komplexer Prozeduren bereitgestellt.

[8] Der Unterschied in der Anzahl der Module in der QuickReference der Dialog Machine (29) und der tatsächlichen Anzahl DM High.DEFs (30) kommt dadurch zustande, dass für die beiden identischen Module DMMathLib und DMMathLF (11 Prozeduren) nur ein Eintrag in der QuickReference existiert.

Tabelle 1: Umfang der RAMSES-Software Die Tabelle gibt Auskunft über folgende Kenngrössen der RAMSES Softwareschichten und wichtiger *Subpackete*: Anzahl Definitions-Module (.DEF), Quelltext in DEFs , Anzahl Implementations Module (.MOD), Quelltext in MODs, Anzahl Binärdateien (.OBM), kBytes der OBMs, Anzahl Bezeichner in DEFs (Konstanten c, Typen t, Variablen v, Prozeduren p und total). In den Angaben über den Umfang des Quelltextes sind Leerzeilen und Kommentare nicht beücksichtigt. Der Umfang der Kommentare kann mit dem Skript in Anhang A.2 separat ermittelt werden. Bei den Analysen wurden alte Versionen (`/.* old.*/`) und die Tests (`/.* Test.*/`) ausgeschlossen. Die Tests sind separat aufgeführt und in der Analyse des gesamten Codebestandes enthalten. Die genauen Auswahlkriterien können dem Script jobs.sh im Anhang A.1 entnommen werden.

Softwareschicht	**.DEF**		**.MOD**		**.OBM**		**Bezeichner in .DEFs**				
	Module	**Zeilen**	**Module**	**Zeilen**	**Module**	**kBytes**	**c**	**t**	**v**	**p**	**total**
RAMSES Software Packete											
DM	91	5'873	122	40'341	62	612	742	475	175	1'715	3'107
DMLibDev	*56*	*5'042*	*67*	*26'193*	*62*	*612*	*714*	*443*	*125*	*1'448*	*2'730*
PCDev/DM	*35*	*831*	*55*	*14'148*	*0*	*0*	*28*	*32*	*50*	*267*	*377*
DMLibDev/High.DEF	*30*	*1'401*	*29*	*16'483*	*31*	*432*	*157*	*84*	*36*	*702*	*979*
RASS	22	1'510	52	15'160	3	20	262	150	33	621	1'066
RASSDev/BatchDMDev	*1*	*6*	*20*	*4'317*	*0*	*0*	*0*	*0*	*2*	*0*	*0*
RASSDev/SysDep	*4*	*126*	*7*	*2'959*	*2*	*16*	*33*	*7*	*4*	*62*	*106*
AuxLib	159	5'457	164	56'444	158	1'316	414	344	204	1'981	2'943
AuxLibDevPD	*32*	*629*	*33*	*5'172*	*33*	*168*	*75*	*13*	*244*	*65*	*397*
AuxLibDevSE	*127*	*4'828*	*131*	*51'272*	*125*	*1'148*	*339*	*279*	*191*	*1'737*	*2'546*
MW	47	2'121	85	21'958	71	484	124	122	332	700	1'278
EMW	10	250	11	6'072	13	168	21	10	45	94	170
ISIS	25	1'720	31	15'388	24	312	237	113	86	543	979
SciLib	56	2'632	55	23'136	54	616	150	109	39	703	1'001
RMS	31	1'309	38	16'540	30	356	184	63	55	380	682
RAMSESDev/PALibDev	*16*	*687*	*16*	*7'624*	*15*	*132*	*6*	*51*	*23*	*184*	*264*
RAMSESDev/RMSLibDev	*15*	*622*	*13*	*6'881*	*15*	*224*	*178*	*12*	*32*	*196*	*418*
RAMSESDev/RMSShellDev	*0*	*0*	*9*	*2'035*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
Tests	58	805	558	84'008	191	1'128	93	36	67	257	453
Total (inkl. Tests)	674	32'412	960	350'528	731	7'264	3'899	2'169	1'282	10'363	17'713
Kompiler, Werkzeuge											
MacMETH	163	5'193	231	100'470	152	1148	291	404	230	2'178	3'103
ToolsDev	36	2'278	86	22'934	29	248	449	167	60	735	1'411

Tabelle 2: RAMSES Schnittstellen Diese Tabelle gibt Auskunft über die Anzahl der von den jeweiligen RAMSES Softwareschichten publizierten Module und deren Bezeichner (Konstanten c, Typen t, Variablen v und Prozeduren p und total). Die Analyse erfolgte auf den QuickReferences, den im RAMSES-Release veröffentlichten Schnittstellen. Diese öffentlichen Schnittstellen, in der DM-Nomenklatur High.DEFs genannt müssen von den Base.DEFs unterschieden werden, deren Verwendung lediglich innerhalb der jeweiligen Softwareschicht vorgesehen ist.

Software-schicht	**Anzahl Module**	**Anzahl Bezeichner in Schnittstelle**				
		c	**t**	**v**	**p**	**total**
DM	29	161	88	44	683	976
AuxLib	72	203	166	71	1'032	1'472
SciLib	30	60	59	11	349	479
MW	7	9	51	11	238	309
ISIS	9	15	38	13	278	344
Total	146	448	402	150	2'580	3'580

3 Kompiler Evaluation

Die Verfügbarkeit eines qualitativ guten und zukunftsträchtigen Modula-2 Kompilers für die Plattform MacOS X/PPC[9] und evtl. für Solaris/SPARC, Windows/x86 und Linux/x86, ist eine Voraussetzung für die Portierung der existierenden RAMSES Software in eine moderne Arbeitsplatzumgebung. Die einzige Alternative dazu ist die Neuimplementation der gesamten Software in einer neue Programmiersprache. Dies könnte entweder manuell, mit Hilfe eines Translators oder durch Kombination dieser zwei Techniken erfolgen.

Die Tabelle 3 zeigt eine Grundauswahl von Modula-2 Kompilern und Translatoren. Die Auswahl ist nicht vollständig, doch ist nach intensiver Internetrecherche davon auszugehen, dass die erfolgsversprechendsten Produkte berücksichtigt wurden. Aufgrund einer ersten Auswahl nach den in der Tabelle aufgeführten Kriterien und nach subjektivem Ermessen wurden die Dokumentationen der im Folgenden diskutierten Kompiler einer eingehenderen Analyse unterzogen.

3.1 GNU Modula-2 Kompiler (GM2)

Der einzige Kompiler der die Zielplattformen Mac OS X, Solaris/SPARC und Linux/x86 in naher Zukunft nativ zu unterstützen verspricht, ist GM2, ein Frontend für die GNU Compiler Collection GCC. GM2 ist zur Zeit der einzige OpenSource Modula-2 Kompiler, der gepflegt und weiterentwickelt wird.[10]

Sprachdefinition: Der GM2 Kompiler ist eine Implementation der Modula-2 Sprachdefinitionen PIM 2-4. Zusätzlich unterstützt er gewisse ISO Modula-2 Eigenschaften und GM2 spezifische Erweiterungen. Diese beinhalten:

- abstrakte Datentypen von irgendeinem (nicht bloss Pointer-Typen)
- Deklarationen in beliebiger Reihenfolge
- Integration von C und Assembler
- Anweisungen für den Präprozessor cpp mit einer Kompiler-Option aktivierbar (Zeilen beginnend mit `#` werden dann als cpp Anweisungen interpretiert)
- Prozeduren und Funktionen mit optionalem Parameter

Funktionsweise des Kompilers: Zur lexikalischen Analyse des Quelltextes benötigt GM2 das weit verbreitete OpenSource Programm flex und einen, beim Bootstrapping selbst erzeugten, rekursiv absteigenden Parser. Grosse Teile von GM2 sind in Modula-2 geschrieben. Für das Bootstrapping benötigt GM2 eine modifizierte Version des Pascal nach C Konverters p2c-1.20.

GM2 ist ein 4-Pass Kompiler. Der erste Pass zerlegt den Quelltext in Tokens und generiert daraus die Symbollisten für die einzelnen Module (Scope Symbols). Die Tokens werden in einem dynamischen Buffer gespeichert. Die 3 folgenden Durchgänge lesen die Tokens aus dem Buffer und generieren Typen und Quadrupels. Das GM2 Frontend übersetzt den Quelltext eines Implementations Moduls in eine Quadrupel-Intermediärsprache, welche dann von der GCC wiederum in den plattformabhängigen Maschinencode übersetzt wird. Die Verwendung der GCC für die Codegenerierung lässt eine hohe Qualität des generierten Codes erwarten.

[9] MacOS X/PPC ist die Hauptplattform der Fachgruppe Systemökologie. Eine Unterstüzung dieser Plattform hat demnach die höchste Priorität.

[10] Siehe http://www.unet.univie.ac.at/ a9406973/modula2/gnum2.html (17. Mai 2004)

Tabelle 3: Grundauswahl Kompiler. Folgende Liste von Modula-2 Kompilern / Translatoren wurde mittels Internetrecherche zusammengestellt. Die Liste ist nicht vollständig. Eine gewisse Vorselektion wurde aufgrund des vom Compiler generierten Codes (Plattform) und aufgrund der Zukunftsperspektiven durchgeführt. Die Referenzen zu den aufgeführten Kompilern sind im Literaturverzeichnis aufgeführt. Die aufgeführten Kriterien haben den Zweck einer Entscheidungsgrundlage für ein weiteres Selektionsverfahren.

Compiler	v	Beschreibung	Sprachdefinition	Werkzeuge	Bibliothek	Plattform
Multi Platform Compiler (Mac OS X PPC, Linux x86, Solaris SPARC)						
Canterbury Modula-2	2.5.35	Modula-2 Translator written in Java, generates Java Source Code	PIM; Object Oriented Language Extensions similar to Oberon-2	Direct import of Java Classes, JDB (Java Debugger), Special Compiler directives for Java, Integrated Make/Build Tool	Modules implemented so far: SYSTEM, InOut,RealInOut, FileSystem, Strings, MathLib0, Conversions, RealConversions	Platform independent, depends on Java 1.1.x or later
GNU Modula-2 GM2 / M2F	0.43	Open Source GCC frontend Compiler, still under development	PIM 2, 3 and 4 fully supported. ISO compatibility under development	Allows Interfaces to C and assembly language. Is highly integrated in the GCC development environment (GCC, gdb, etc.). cpp Preprocessor Invocation. Compiler Frontend m2f	ISO Library partially implemented, needs to be completed. PIM libraries are very mature.	Generates native Code for UNIX/Linux Platforms on the Architectures i386 and SPARC. OS X may be supported in future. Cross-Compilation for Strong ARM Processors and for Windows Plattforms
Stony Brook M2 Compiler	b28	A maintained compiler for the x86 and SPARC Environment. Extensions teilweise Kompatibel mit P1	ISO compliant with several extensions (dynamic arrays, conditional compilation, etc.)	Editor, Debugger, Profiler: Integrated Development Environment	ISO Library with lots of extensions (e.g. platformspecific)	32-Bit IA Windows and Linux, 32-Bit SPARC Solaris
Gardens Point Modula	-	Free high developed Modula-2 compiler and environment for different platforms. At least partially maintained	Seems to use a own language definition, with slight differences to PIM-2 and ISO	Compiler, Linker, Profiler, Compilation Analysys tool, Cross Reference Generator	A huge Library with Standard Libraries ISO, PIM-2 and special Libraries	Linux, Windows (Cygwin), FreeBSD, SPARC,

Mocka	9905	Open Source Karlsruhe Modula-2 Compiler, still maintained? (last version 1999)	PIM-3	Interactive session mode allows choice of editor. Error messages shown in the source. Optimizer, gdb support, make facility	Standard Libraries, FOREIGN MODULE allow integration of other languages	Native on different UNIX/Linux platforms (Sun, Linux, etc.). Free and Open Source for Linux. C-backend for portable C code
Mac OS X PPC						
P1 Modula	7.3	Well maintained Modula-2 Environment for the Macintosh Computer	ISO 10514 compliant (incl. ext 2 and 3) Language extensions like mac, dynamic arrays;	Powerfull Pragma System, Language integration (C, C++, Assembler, Pascal). Source level debugging	ISO Library; Macintosh Universal Interfaces	Integrated in MPW. Supports 68k and PowerPC Mac. Interfaces Carbon Libraries, Native OS X / Cocoa support is planned.
Linux x86						
Native XDS-x86	2.51	Compiler for Modula-2 and Oberon-2. Calls itself the most advanced optimizing ISO Compiler on the market. (Single Pass!)	ISO 10514 compliant; PIM	Access to the OS API, Allows Integration of Modula-2, Oberon-2, C and Assembly Code.	Most (but not fully) ISO Library. Runtime Library supports Garbage Collection, Postmortem History, etc.	Versions native for Windows and Linux on x86 Plattforms
XDS-C		Modula-2 and Oberon-2 Cross Programming System, generates portable C Code	See Native XDS-x86	See Native XDS-x86	See Native XDS-x86	See Native XDS-x86, Porting can also be done by Excelsior. Migration Services also offered.
Solaris SPARC						
EPC Modula-2	2.0.8	The maintaining company Edinburgh Portable Compiler (EPC) was taken over and so, the compiler isn't maintained anymore.	PIM 3 and 4; SUN Dialect (REAL compatible with LONGREAL)	Tools and Extensions described in the EPC Modula-2 Reference Manual	Libraries described in the EPC Modula-2 Reference Manual	Generates Binary Code for SUN Workstations with a SPARC Architecture
Ulm's Modula-2 Compiler	3.0b8	OpenSource Modula-2 Compiler for SUN Computer, still maintained?	PIM3, PIM1, Wirt94. PIM4 and ISO will never be supported	Makefile Generator	? Manpages available on the Internet	SPARC/Solaris 68k/SunOS

Der GM2 Kompiler wird in der UNIX Shell, d.h. über die Kommandozeile ausgeführt. Nebst den Standard GCC Kompiler Optionen stehen verschiedene GM2 spezifische Optionen zur Verfügung. Ein Teil davon ist in der Tabelle 4 aufgeführt.

Wie die Tests mit GM2 zeigten, hat der Kompiler einige sehr fortschrittliche Eigenschaften. Ausgehend von einem Implementations Modul kompiliert er automatisch alle importierten Module (rekursiv absteigend), sofern diese nicht schon in der aktuellsten Version als Binärdatei vorliegen und linkt diese zu einem ausführbaren Programm zusammen. Der GM2 benötigt für den Import von Schnittstellen auch keine Symboldateien, sondern bezieht sich direkt auf die jeweiligen Definitions-Module.

Werkzeuge (Linker, Debugger) und deren Funktionsweise: GM2 profitiert nebst der Qualität und Popularität der GCC auch von deren Entwicklungsumgebung. Es gibt eine Menge Tools, welche rund um die GCC entwickelt wurden. So z.B. GNU Debugger (GDB), GNU Profiling (GPROF), GNU Coverage (GCOV), Emacs Editor, Make, Configure, etc. Dabei handelt es sich fast ausschliesslich um befehlsbasierte Terminal-Programme ohne grafische Benutzerschnittstelle. Einzig der Editor Emacs ist auch in einer X11-Version erhältlich. Eine funktionell umfangreiche Integration der gesamten Entwicklungsumgebung in den Emacs Editor ist zu erwarten. Inwiefern andere Programme wie z.B. der Editor Alpha eine Integration der GCC ermöglichen, müsste abgeklärt werden.

Bibliothek: Die Bibliothek von GM2 beinhaltet sowohl PIM wie auch ISO konforme Bibliotheken. Auch die Standardfunktionen und -prozeduren sind sowohl in der Version PIM wie auch in der Version ISO vorhanden. Der Kompiler selbst wird mit Hilfe von PIM Bibliotheken und dem Aufruf gewisser C Bibliotheken erstellt. Die PIM Bibliotheken sind sehr ausgereift, wird doch vom Kompiler selbst extensiv von ihnen Gebrauch gemacht. Viele der Bibliotheken nach ISO-Definition sind ebenfalls implementiert. Die ISO Bibliothek ist jedoch nicht komplett und nicht so gut getestet, wird jedoch laufend ergänzt und verbessert. Hinzu kommen noch ein paar GM2 spezifische Erweiterungen. Von grösster Tragweite ist die Enführung des Typs `String` im Modul `DynamicStrings` was eine Reihe äquivalenter Module, für den Umgang entweder mit `Array Of Char` oder eben mit `String`, zur Folge hat.

Das mit dem GM2 Kompiler ausgelieferte Werkzeug h2def, verspricht eine halbautomatische Übersetzung von C Header Dateien in Modula-2 Definitions Module.[11] Von dem her besteht im Prinzip die Möglichkeit, sich sämtliche C Bibliotheken zugänglich zu machen. Insbesondere bietet sich h2def auch zur Übersetzung von Schnittstellen zum Betriebssystem an (z.B. CarbonLib oder Cocoa). Da das Werkzeug selbst vom Kompiler unabhängig ist, sollte es auch zusammen mit einer anderen Modula-2 Implementation verwendet werden können.

Portabilitätseigenschaften: Momentan läuft GM2 nur unter Linux, jedoch auf einer Vielzahl von Rechnerarchitekturen. Der Mailingliste zufolge ist auch ein Bestreben im Gange, GM2 auf die Plattformen Solaris/SPARC und OS X/PPC zu portieren[12], was dank der weiten Verbreitung der GCC keine allzugrossen Probleme bereiten sollte.[13]

Laut Angaben auf der Homepage von GM2 soll es auch möglich sein, Code für die Windows-Umgebung mittels der Cross-Kompiler Funktionalität von GCC zu erzeugen.

Testkompilation: Erste Testkompilationen mit dem GM2 Kompiler sind ohne grössere Probleme verlaufen. Folgende Probleme wurden festgestellt:

[11] Laut dem Handbuch von GM2, kann h2def nur triviale C Header Files übersetzen. Insbesondere versteht h2def keine C Präprozessor-Anweisungen. In einem solchen Fall muss eine manuelle Intervention erfolgen.

[12] Eintrag Mailingliste: http://floppsie.comp.glam.ac.uk/pipermail/gm2/2004-May/000255.html (17. Mai 2004)

[13] Da die Codegenerierung des GM2 Kompilers von der GCC übernommen wird, sollte das Frontend GM2 selbst relativ leicht portierbar sein.

- `BITSET` muss aus dem Modul `SYSTEM` importiert werden
- Die Bezeichnung von Parametern muss in den Defintions- und in den Implementations-Modulen übereinstimmen. Das ist der am häufigsten aufgetretene Fehler.

V.a. der zweite Punkt könnte im Falle einer Portierung eine Menge Arbeit geben, da an vielen Stellen in der RAMSES Software die Parameter von Funktionen und Prozeduren in Definitions- und Implementations-Modulen unterschiedlich bezeichnet werden. Zusätzlich hätte eine solche Anpässung unerwünschte Effekte auf die Eleganz und die Lesbarkeit des Quelltextes. Oft werden in Definitions-Modulen aus Dokumentationsgründen sehr lange und aussagekräftige Bezeichner gewählt, während in Implementations-Modulen eher mit kurzen Bezeichnern gearbeitet wird. In Implementations-Modulen kann es auch zu Konflikten mit lokalen Variablen gleichen Namens kommen.

Portierungsaufwand: Wie die ersten Kompilationsversuche mit GM2 gezeigt haben, sind dank der PIM-Kompatibilität bezüglich Sprachdefinition keine allzugrossen Probleme bei der Portierung der RAMSES Software zu erwarten.

Mehr Probleme dürften mit dem Fehlen plattformspezifischer M2-Bibliotheken wie z.B. Universal Library, bzw. CarbonLib auf der Plattform MacOS X/PPC auftreten. Probleme mit Bibliotheken sollten jedoch ohne allzugrosse Probleme über C-Schnittstellen gelöst werden können.

Allgemeines und Gesamteindruck: Mit der Version Version 0.43 befindet sich der GM2 Kompiler noch in einer frühen Entwicklungsstufe. Wie obige Besprechungen und die Tabelle 4 zeigen, konnte jedoch ein grosser Teil der folgenden Projektziele bereits realisiert werden.

- Modula-2 Frontend für GCC mit Sprachunterstützung PIM-[234] und dem ISO Standard in zweiter Priorität
- PIM Bibliotheken Kompatibilität
- ISO Bibliotheken Kompatibilität (noch vor der ISO Sprachunterstützung des Kompilers)
- Integration weiterer Bibliotheken (Topspeed, Logitech, Ulm)
- Einfache Schnittstelle zu C
- Fester Bestandteil der GCC Standarddistribution
- Ausnützung aller Eigenschaften der GCC
- Benutzerbedürfnisse efüllen

Trotz seiner jungen Geschichte macht GM2 einen ziemlich ausgereiften Eindruck. Dies kommt wohl daher, dass GM2 einerseits auf dem Modula-2 Kompiler m2f beruht, von welchem ein grosser Teil des Quelltextes übernommen werden konnte und andererseits durch den Einbezug erfolgreicher Projekte aus dem OpenSource Umfeld (p2c, flex, gcc, gdb, emacs, etc.) grosse Teile der Funktionaltät auslagern kann.

Die von GM2 verwendeten Programme überzeugen nicht nur bezüglich ihrer Qualität sondern auch bezüglich ihrer Verbreitung. Insbesondere die GCC (aktuelle Version 3.4.0) ist eine beständige und qualitativ hoch entwickelte Entwicklungsumgebung und hat sich, wie die Portierung auf mittlerweile fast alle Plattformen zeigt, zu einem Quasi-Standard entwickelt. So ist die GCC (mit dem Objective C Frontend) mittlerweile auch die Standard-Entwicklungsumgebung auf den neuen Apple Plattformen.

Die Offenlegung des Quelltextes, die Schlankeit von GM2 aufgrund der Integration verschiedener Programme sowie die Renommiertheit dieser Programme selbst, weisen auf eine vielversprechende Zukunft dieses Modula-2 Kompilers hin. Diese Hoffnung wird einzig durch die Tatsache getrübt, dass sich momentan eigentlich nur eine Person (Gaius Mulley) die Entwicklung von GM2 vorwärts treibt und Modula-2 als Programmiersprache kaum mehr das Potential hat eine grosse Entwicklergemeinde zu mobilisieren.

Die Verwendung desselben Kompilers für mehrere, im Idealfall alle, Zielplattformen würde grosse Vorteile mit sich bringen: Geringerer Aufwand für die Kompilerwartung und den Unterhalt verschiedener Code-Versionen (Ports), Vergleichbare Performanz und Codequalität auf allen Plattformen. Dies steht im Moment noch nicht zur Diskussion, da GM2 zur Zeit erst für diverse Linux Umgebungen zur Verfügung steht.

3.2 P1 Kompiler

P1 ist ein auf MPW[14] basierter Modula-2 Kompiler für die Macintosh Plattform. Für den im Juni 2004 erscheinenden Prerelease 8.0a1 ist eine native OS X Unterstützung (Integration in XTools, calling conventions und linking standards des Mach-O Kernels) mit voller Unterstützung der Carbon Bibliothek geplant (inkl. Erweiterungen sie wie in der Carbon Dokumentation welche mit XTools ausgeliefert wird definiert wurden). Die Version 8.0 soll im Oktober auf den Markt kommen.

Die folgende Beschreibung des Kompilers bezieht sich auf die aktuelle Version 7.3. Ausblicke auf die kommende Version 8 und folgende werden jeweils explizit erwähnt.

Sprachdefinition: Die P1 Modula-2 Implementation basiert auf dem ISO-Standard IS 10514. Neben der Basis Sprachdefinition 10514-1 unterstützt P1 auch die beiden Sprach-Erweiterungen IS 10514-2 (Generics) und 10514-3 (Object Oriented Modula-2). Weiter werden die ISO Standards von P1 spezifischen Erweiterungen ergänzt:

- Mac spezifische Typen, Funktionen und Prozeduren in SYSTEM (siehe Tab. 4)
- Kompileranweisungen im Quelltext (Pragmas) u.a. für konditionelle Kompilation
- Einbindung (Interfacing) der Sprachen C, Pascal, 68k System Calls (Traps)
- Möglichkeit zur Einbindung von M2 Code in andere Sprachen
- Dynamische Arrays
- AltiVec Befehlssatz

Funktionsweise des Kompilers: P1 ist ein 3-Pass Kompiler. Wie genau die Übersetzung des Quelltextes vor sich geht konnte nicht ausfindig gemacht werden. Auch ist nicht klar, in welche Intermediären Sprächen der Quellcode übersetzt wird. Die Tatsache, dass der P1 Kompiler Modula-2 Quelltext nach C/C++ übersetzen kann, lässt vermuten, dass der Quellcode auch Kompiler-Intern in eine C-artige Intermediärsprache übersetzt wird. Genauere Angaben dazu wurden keine gefunden. Für die Version 8 des P1 hat Herr Wiedemann, der Entwickler von P1 angekündigt, die Codegenerierung via Assembler zu machen, bzw. den von P1 generierten Assemblercode von XTools, der MacOS X Entwicklungsumgebung, in Maschinencode übersetzen zu lassen.

[14] **MPW: M**acintosh **P**rogrammer's **W**orkshop

Diese Integration von P1 in XTools steht jedoch noch bevor. Mit der Integration der aktuellen Version P1 7.3.6 in MPW, ist P1 nach wie vor auf die Classic Umgebung angewiesen. Jedoch können schon heute Programme für die OS X Umgebung geschrieben werden. So unterstützt P1 in der Codegenerierung nebst den 68k Prozessoren (mit SANE Instruktionen) auch die PPC Architektur (inklusive AltiVec Instruktionen moderner G4 Prozessoren).

Werkzeuge (Linker, Debugger) und deren Funktionsweise: Die aktuelle Version des P1 benötigt den MPW Linker link, bzw. ppclink. Hingegen beeinhaltet er eine eigene symbolische Debugging Umgebung auf dem Quelltext Level. Das zuletzt ausgeführte Statement wird in einem Fenster angezeigt, Listen, Bäume und andere Pointer können per Doppelklick analysiert werden. Auch der Speicher-Inhalt kann in verschiedenen Darstellungen angeschaut werden. Der Debugger unterstützt Breakpoints und Einzelschritt-Breakpoints.

P1 bietet weiter zwei Tools GenMake und M2Cross. GenMake generiert automatisch MakeFiles für P1 Modula-2 Programme, M2Cross analyisiert die Abhängigkeiten zwischen den Modulen und erstellt eine globale Cross-Reference Tabelle mit allen exportierten Bezeichner.

Bibliothek: Die von P1 zur Verfügung gestellte Bibliothek umfasst nebst der Standard ISO Bibliothek gewisse Macintosh spezifische Bibliotheken und Schnittstellen. Die P1 spezifischen Erweiterungen der ISO Bibliothek beinhalten unter anderem Module für zusätzliche Dateisystemfunktionalitäten, I/O zusätzlicher Datentypen und Zugang zum MPW. Folgende Schnittstellen zum API des Macintosh Betriebssystem werden von P1 zur Zeit unterstützt: die aktuellsten Universal Libraries 3.4.2 (nur Classic), Carbon (Classic und OS X). Bezüglich der Vollständigkeit der Carbon Schnittstelle konnte nichts genaueres in Erfahrung gebracht werden. Es ist jedoch anzunehmen, dass sie vollumfänglich in Form von "Foreign Definition Modules" angeboten wird[15]. Weiter wird für die P1 Version 8.1 eine Schnittstelle zum Cocoa API in Aussicht gestellt.

Portabilitätseigenschaften: Der P1 Kompiler beschränkt sich in seiner Anwendbarkeit auf die Macintosh Plattform. Einzig mit der Übersetzungsfunktionalität nach C/C++ liesse sich der für P1 angepasste Quelltext auch auf andere Plattformen portieren. Ob diese Möglichkeit zur Generierung von C-Quelltext jedoch auch mit der neuen Version 8 bestehen bleibt, ist fraglich[16].

Dahingegen scheint P1 die Portierung von Modula-2 Software innerhalb der Mac Plattform (von 68k nach PPC und von Classic nach OS X) hervorragend zu unterstützen. Da P1 die Codegenerierung sowohl für 68k wie auch für PPC Prozessoren unterstütz, sowie das Linken von Programmen mit reinen Classic (Universal Libraries) und Carbon Bibliotheken ermöglicht, können Programme für alle Mac Plattformen (Classic/68k, Classic/PPC und OS X/PPC) kompiliert werden. Die Konsequenz davon wäre, dass man denselben Kompiler und evtl. denselben Quelltext verwenden könnte für sämtliche Mac Plattformen. Bedingung dafür wäre jedoch die Migration aller 68k- und MacOS 9 Toolbox-Anweisungen auf die Mac OS X Carbon Bibliothek Aufrufen im Quelltext.

Testkompilation: Umfangreiche Testkompilationen wurden mit einer Vorabversion von P1 8.0 durchgeführt. Die Resultate schienen vielversprechend, weswegen P1 für eine vertiefte Analyse ausgewählt wurde. Die Resultate der Testkompilationen sind im Kapitel 4 zu finden.

[15] Alle in den Tests des P1-Kompilers verwendeten Aufrufe der Carbon Schnittstellen funktionierten einwandfrei.

[16] Ab Version 8 soll der P1 Kompiler in die XTools-Entwicklungsumgebung von OS X integriert werden. XTools verwendet eine Assemblersprache als intermediären Code. Aus Kompatibilitätsgründen könnte jedoch die Übersetzungsfunktion nach C erhalten bleiben.

Es ist zu beachten, dass sich während dieser Arbeit der P1 Kompiler wesentlich verändert hat. Dieser Beschrieb des Kompilers basiert jedoch auf der Dokumentation der Version 7.3.6. Änderungen des Kompilers wurden nicht nachgeführt und müssen der aktuellen Dokumentation entnommen werden.

Portierungsaufwand: Die Portierung der Mac Version RAMSES Software von MacMETH auf den P1 Kompiler und von der Classic/68k auf die CarbonLib/PPC Plattform umfasst eben diese beiden Schritte: a) die Anpassung des Quelltextes an die Sprachdefinition und die Bibliotheken des P1 Kompilers für die Portieurng der BatchDM und b) das Umschreiben der 68k Befehle (System Calls) bzw. der Toolbox-Aufrufe (Universal Libraries Calls) in CarbonLib-Aufrufe für die Portierung der interaktiven RAMSES Simulationsumgebung mit grafischer Ausgabe.

Der erste Schritt müsste sicher darin bestehen, den Quelltext den Spezifikation des P1 Kompilers anzupassen. Der bei Testkompilationen am häufigsten aufgetreten Fehler beruht auf der Auslagerung der Typen LONGCARD und LONGINT aus dem Set von Standardtypen in das Pseudo-Modul SYSTEM. Dieses Problem ist relativ einfach durch eine zusätzliche IMPORT Anweisung zu beheben. Ein weiterer regelmässig auftretender Fehler kommt durch die unterschiedliche Handhabung der Typen Konversion zustande.

Die vorzunehmenden semantischen und allenfalls syntaktischen Anpassungen des Quelltextes können wohl grösstenteils mit Hilfe eines Skripts automatisiert werden.

Grössere Probleme könnten mit der Portierung der Schnittstelle zum System auftreten. Bisherige Aufrufe der Toolbox (Universal Libraries) können wohl zum grössten Teil 1:1 übernommen werden, da dieselben Definitionen Prozeduren auch von der CarbonLib angeboten werden. Von der CarbonLib nicht unterstützt werden sämtliche 68k spezifischen Funktionen, Funktionen welche Heap-Speicher allozieren, Funktionen welche direkt auf die Hardware zugreifen, sowie direkter 68k Code und der Zugriff auf die Trap-Tabelle und den Patch Manager (da ebenfalls 68k spezifisch).

Für die Portierung eines Classic Programmes auf die Schnittstellen der CarbonLib schlägt Apple folgendes Vorgehen vor:

1. *PPC nativer Quelltext:* Sämtlicher 68k spezifische Quelltext muss entfernt und gegebenenfalls durch entsprechende Aufrufe der Universal Libraries ersetzt werden. Dies betrifft einerseits direkte 68k Instruktionen, andererseits 68k spezifische Toolbox-Aufrufe.

2. *Aktuellste Universal Libraries:* Die Portierung nach Carbon wird durch die Verwendung der aktuellsten Universal Libraries (Version 3.4.2) wesentlich vereinfacht.

3. *Transition nach Carbon:* Lässt sich das Programm unter Verwendung der aktuellsten Universal Libraries für die PPC Plattform kompilieren, sollte der Wechsel nach Carbon nicht mehr allzugrosse Schwierigkeiten bereiten.

Alle diese, für eine nahtlose Carbonisierung benötigen Schritte, werden vom P1 Kompiler hervorragend unterstützt und sollten deshalb mit einem minimalen Aufwand bewältigt werden können.

Allgemeines und Gesamteindruck: Eine Demo-Version des P1 Kompilers v 7.3 ist auf der Homepage verfügbar. Jedoch ist diese auf die Generierung von PPC Code limitiert, hat einen limitierten Namensraum (max. 1931 Bezeichner) und beeinhaltet keinen Quelltext der Universal Interfaces und keine Carbon Schnittstellen. Die Vollversion kostet ca. sFr. 500.-.

I.A. macht der P1 Kompiler einen sehr ausgereiften Eindruck und würde sich auf Grund seiner Geschichte für eine nahtlose Portierung der RAMSES Software von Mac OS nach Mac OS X eignen. So werden z.B. die Plattformen 68k und PPC mit ihren Eigenheiten (z.B. SANE, AltiVec) vollumfänglich unterstützt.

Der Nachteil dabei ist die starke Bindung des Kompilers an die Mac Plattform. Für die UNIX bzw. Windows Ports müssten andere Kompiler verwendet werden. Die verschieden Ports müssten demnach separat gewartet werden.

Wie der GM2 Kompiler, wird auch der P1 Kompiler von ein paar wenigen Entwicklern vorangetrieben. Im Gegensatz zum GM2 Kompiler handelt es sich nicht um ein OpenSource Projekt, d.h. es ist nicht damit zu rechnen, dass sich das Entwicklerteam mit der Zeit vergrössern wird. Das bedeutet ein gewisses Risiko, was die zukünftige Entwicklung des Kompilers anbelangt. Da der SourceCode nicht veröffentlicht wird besteht auch nicht die Möglichkeit, den Kompiler selbst zu warten. Herr A. Wiedemann, der Entwickler des P1-Kompilers hat jedoch in einer persönlichen Unterhaltung angeboten, den Quelltext im Notfall zu veröffentlichen. Zudem scheint Herr Wiedemann für eine ausgezeichnete Pflege des Kompilers bekannt zu sein.

3.3 XDS-C

XDS-C ist insofern ein Translator und kein eigentlicher Kompiler, dass er keinen Maschinencode erzeugt, sondern Modula-2 Quelltext lediglich in C/C++ Quelltext übersetzt, welcher anschliessend mit einem C Kompiler in Maschinencode übersetzt werden kann.

Sprachdefinition: XDS-C unterstützt die ISO 10514 Sprachdefinition und verfügt über eine Menge an Erweiterungen, welche über Kompileranweisungen aktiviert werden können. Es wird auch die Übersetzung von Aufrufen von C Libraries und System APIs unterstützt.

Bibliothek: XDS-C selbst umfasst eine fast vollständige ISO 10514 und PIM Bibliothek, Schnittstellen zu den Standard ANSI C Bibliotheken und bietet selbst noch Erweiterungen an. Ebenfalls verfügt XDS-C über ein Programm zur Übersetzung von C header files in Modula-2 Definitions Module.

Portabilitätseigenschaften: XDS-C läuft auf den Plattformen Linux/x86 und Windows. Der resultierende C Quelltext sollte jedoch Plattform unabhängig sein und sich auf allen Plattformen mittels einem nativen C Kompiler, z.B. dem gcc, in Maschinencode übersetzen lassen. Der von XDS-C generierte Quelltext ist jedoch kaum lesbar und eignet sich nicht für eine manuelle Wartung verschiedener Ports. Es müsste also weiterhin der bestehende Modula-2 Code gewartet werden, welcher für jeden Release erst durch XDS-C in C/C++ übersetzt und anschliessend mit einem C Kompiler compiliert werden müsste. Dieser indirekte Weg vom Quelltext zum Maschinencode über eine intermediäre Sprache wird jedoch mittlerweile aus Flexibilitätsgründen auch von vielen eigentlichen Kompilern verwendet. So generiert z.B. auch der P1 Kompiler einen intermediären Assemblercode, welcher anschliessend entweder von P1 selbst oder von XTools in Maschinencode übersetzt und gelinkt wird.

Portierungsaufwand: Der Portierungsaufwand und auch die Vorgehensweise bei der Portierung ist wohl mit dem GM2 Kompiler identisch, vorausgesetzt das GM2 Frontend ist für alle Zielplattformen vorhanden. Vom Kompiler werden in etwa dieselben Libraries zur Verfügung gestellt (C Libraries, Win32 API). Andere Schnittstellen (v.a. Toolbox, Carbon unter OS X) müssten in beiden Fällen selbst erzeugt werden. XDS-C und GM2 liefern dazu beide ein Tool h2def, welches C Header Dateien in M2 Definition Modules übersetzt. Wo XDS-C intermediären C-Code erstellt, wird bei GM2 direkt der GCC für die Generierung des Source-Codes verwendet.

Allgemeines und Gesamteindruck: Die vertreibende Firma Excelsior hat wahrscheinlich viel Erfahrung in der Portierung von M2-Software. Sie verwenden das Produkt selbst für solche Aufgaben. Es

macht jedoch der Anschein, dass die M2 Kompiler nicht mehr zu ihren Kernprodukten gehören. Mehr ist über die Zukunftsträchtigkeit nicht in Erfahrung zu bringen. Auch hier stellt sich demnach das Problem, dass die Weiterentwicklung der Kompiler nicht garantiert ist und dieser, da es sich nicht um eine OpenSource Projekt handelt auch nicht selbst gewartet werden könnte. Der Preis für XDS-C beläuft sich auf ca. sFr. 1500.-

3.4 Gardens Point Modula-2 (GPM)

Sprachdefinition: Der Kompiler basiert auf dem ISO Standard 10514, implementiert diesen jedoch nicht vollständig.

Werkzeuge (Linker, Debugger) und deren Funktionsweise: GPM bringt eine umfangreiche Entwicklungsumgebung mit sich.

Bibliothek: GPM stellt eine umfangreiche Bibliothek zur Verfügung (ISO fast komplett, PIM teilweise, eigene Ergänzungen für UNIX Umgebung).

Portabilitätseigenschaften: GPM ist ein Modula-2 Kompiler für die Plattformen Linux/x86, Sparc und Windows. Von GPM gibt es jedoch keinen Mac OS / OS X Port, so dass dieser Kompiler höchstens in Kombination mit einem Kompiler für OS X (P1, GM2 oder XDS-C) verwendet werden kann. Da GM2 und XDS-C auch Code für die Plattformen Linux/x86 und Windows generieren können (ein Kompiler für alle Plattformen), kommt der GPM eigentlich nur in Kombination mit P1 in Frage. Aber auch da scheint der GPM, sofern es keine driftigen Gründe gibt, nicht die ideale Wahl zu sein, denn der Windows Port stützt sich auf der .NET-Technologie ab. Für das RAMSES-Entwicklerteam ist diese Technologie weitgehend unbekannt und da es sich bei Windows um eine Sekundärplattform handelt ist wohl auch das Interesse nicht allzugross, sich dieses Wissen anzueignen.

Allgemeines und Gesamteindruck: Der sehr umfangreichen Dokumentation ist zu entnehmen, dass der Kompiler hoch entwickelt und sehr schnell ist. Da die Dokumentation zu umfangreich ist für eine eingehendere Analyse und wichtige Informationen nicht in einer Übersicht hervorgehoben werden, wird auf eine detailierte Auflistung der Eigenschaften von GPM in der Tabelle 4 verzichtet.

Auf der Homepage des GPM macht es den Eindruck, dass der Kompiler v.a. den 90er Jahren entwickelt wurde. Es werden jedoch auch heute noch neue Releases veröffentlicht (z.B. Sparc Jan 03) und sogar neue Ports entwickelt. So sind die CLR Versionen für die Windowsplattform relativ neu und stützen mit Hilfe eines intermediären Codes auf der .NET Technologie von Microsoft ab.

GPM ist ein wissenschaftliches Projekt und ist gratis erhältlich. Der Source Code ist jedoch nicht offen verfügbar. Bisher haben viele Leute an der Entwicklung des Kompilers mitgeholfen. Es macht jedoch den Anschein, dass das Projekt von der Gruppe nicht mehr stark vorangetrieben wird. Es wäre vorstellbar, da es sich um ein aus öffentlichen Geldern bezahltes Projekt handelt, dass der Source Code bei einer Aufgabe des Projektes für die Pflege übernommen werden könnte.

3.5 Mocka

Portabilitätseigenschaften: Der Mocka Kompiler beschränkt sich auf die Zielplattformen Solaris/SPARC und Linux/x86. Insofern ist Mocka nur dann interessant, wenn man für die Hauptplattform Mac OS X einen separaten Kompiler verwendet und auch für den Windows-Port nach einer eigenen Lösung sucht.

Mocka bietet jedoch auch die Möglichkeit, einen intermediären C-Code zu erzeugen, welcher dann auf den verschiedenen Plattformen nativ compiliert werden könnte. Sollte man sich für ein Vorgehen in dieser Art entscheiden, müsste nebst dem XDS-C Translator auch der Mocka Kompiler in Betracht gezogen werden.

Allgemeines und Gesamteindruck: So wie es aussieht wird der Kompiler nicht mehr intensiv gewartet. Der letzte Linux Release stammt aus dem Jahre 1999. Auch macht es nicht den Anschein, dass der Kompiler noch auch weitere Plattformen portiert werden sollte. Deshalb wird in Folge nicht mehr näher auf den Mocka Kompiler eingegangen.

Tabelle 4: Eingehendere Kompileranalyse. Die Dokumentationen der drei vielversprechendsten Kompiler GM2, P1 und XDS-C wurden gemäss einer Kriterienliste durchsucht und die Resultate in folgender Tabelle zusammengestellt.

Kriterium	GM2	P1	XDS-C
Formale Sprachkriterien			
Sprachdefinition	PIM-[234], ISO Erweiterungen und eigene Erweiterungen: Abstrakte Datentypen können beliebigen Typ annehmen, Deklarationen in beliebiger Reihenfolge, Schnittstelle zu C und Assembler, Preprocessoranweisungen, optionale Parameter, etc.	ISO 10514-[123], Objektorientiert (10514-3)	ISO 10514 und eigene Spracherweiterungen: Zusätzliche numerische Typen (SHORTINT, LONGINT, SHORTCARD, LONGCARD), Casting, BYTE Zuweisung, Dynamische Arrays, Konstante Arrays (inkl NEW und DISPOSE), Set Komplement, Read-Only Parameter, variable Anzahl Parameter, Read-Only Export, Umbenennung importierter Module, HALT, ASSERT
Open array		wird unterstützt	unterstützt dynamic arrays
Standardfunktionen	HIGH, CAP, ABS, ODD, VAL, CHR, MIN, MAX, LENGTH (ISO)	Funktionen (ABS, CAP, CHR, CMPLX, FLOAT, HIGH, IM, INT, ISMEMBER, LENGTH, LFLOAT, MAX, MIN, ODD, ORD, RE, SIZE, TRUNC, VAL)	Funktionen (ABS, ASH, CAP, CHR, CMPLX, ENTIER, FLOAT, HIGH, IM, INT, LEN, LENGTH, LFLOAT, MAX, MIN, ODD, ORD, RE, SIZE, TRUNC, VAL), Prozeduren (ASSERT, COPY, DEC, DISPOSE, EXCL, HALT, INC, INCL, NEW)
Pragmas	s. o. Sprachdefinition	Viele Pragmas (Compileranweisungen). U.a.: verschiedene Tests, AltiVec Extension, Floating Point Optimization, Foreign Module, PPC/68k Codegeneration, Conditional Compilation, Definition for C Pragma for C-Calling convention	Inline (PUSH, POP, ASSERT, CHECKINDEX, CHECKDIV, CHECKINDEX, CHECKNIL, CHECKPROC, CHECKRANGE, CHECKSET, GENTYPEDEF, NOEXTERN, WOFF, ALIGNMENT, ENUMSIZE, SETSIZE), Header (CHANGESYM, COMMENT, CSTDLIB, GCAUTO, GENCDIV, GENCPP, GENCONSTENUM, GENCTYPES, GENDEBUG, GENHISTORY, GENKRC, GENPROCLASS, GENPROFILE, GENSIZE, LINENO, M2ADDTYPES, M2BASE16, M2EXTENSIONS, NOHEADER, STORAGE), Conditional Compilation.

Spracheinbindungen	"DEFINITION MODULE FOR "C", Inline Assembler wie in C: ASM VOLATILE	Schnittstellen zu C, Pacal, 68k Traps (Foreign Code Modules), Schnittstellen zu M2 (External M2 Calls).	Unterstützt Implementationen in den Sprachen Oberon-2, Modula-2, C, Pascal, Win32 API und OS/2 API. Schnittstellen zu C werden unterstützt (direct calls, foreign definition modules)

Funktionalität und Qualität des Compilers			
Grundeigenschaften	4-Pass Compiler, grösstenteils in Modula-2 geschrieben. Bootstraping mit p2c-1.20. Generiert Symboltabelle und GCC-Baum. Generierung Maschinencode durch GCC.	3-Pass Compiler, Kann Maschinencode, Assemblercode oder C/C++ Code generieren.	1-Pass Compiler, d.h. Bezeichner müssen vor Gebrauch deklariert werden (FORWARD Deklaration). Generiert C/C++ Code.
Datentypen	INTEGER, LONGINT, CARDINAL, LONGCARD, BOOLEAN, REAL, LONGREAL, SHORTREAL und CHAR	Anpassung der Typen an 68k (16-bit word) und PPC (32-bit word) mit Compileranweisungen (Pragmas). Zusätzliche eigene Erweiterungen in SYSTEM	Standardtypen und Erweiterungen (siehe Sprachdefinition und SYSTEM)
SYSTEM	ISO Standard Konstanten: (LOCSPERBYTE, LOCSPERWORD, BITSPERLOC), ISO Standard Typen (LOC, ADDRESS, BYTE, WORD) und ISO Standard Prozeduren (ADDADR, SUBADR, DIFADR, MAKEADR, ADR, CAST, ROTATE, SHIFT, TSIZE)	Zusätzlich zum ISO Standard: Typen (INT[8,16,32], CARD[8,16,32], WORD[8,16,32], SET[8,16,32], LONGINT, LONGCARD, ADRCARD, EXTENDED, EXTENDEDCOMPLEX, LONGDOUBLE, DOUBLECOMPLEX, BCD, STR255), Prozeduren (REGISTER, SETREGISTER, CODE, INCADR, DECADR, TOSTR255, FROMSTR255, BREAK, SETEXITCODE, ROUND, LONG, SHORT, CLONE, OFFS, ACCESSGLOBALS)	Zusätzlich zum ISO Standard: Typen (INT[8,16,32], CARD[8,16,32], BOOL[8,16,32], INDEX, DIFADR_TYPE, INT, CARD), C Typen (int, unsigned, site_t, void), Prozeduren (MOVE, FILL, GET, PUT, CC)
Data alignment		Pragma für 68k und PPC Alignment	ALIGNMENT Pragma (<* ALIGNEMENT = [1,2,4,8] *>)
Linken		GCC / ld (P1 v-8.x)	Linken ist Sache des verwendeten C-Compilers
Komfort Entwicklung	Integration in ausgereifte UNIX Entwicklungsumgebung. Tools für effizentes Interfacing (h2def: .h − > .DEF)	Integration in Mac Entwicklungsumgebung MPW. Debugging auf Source Level.	Viele Bibliotheken und Schnittstellen. Tools wie h2def. Keine eigene und auch keine spezielle Integration in bestehende Entwicklungsumgebung.
Korrektheit			
Präzision		IEEE Double Precision, SANE (68k), AltiVec (PPC)	

Optimierung / Laufzeit		Compiler Anweisung für Optimierung i.A. und Float Arithmetik	Der statische OSA Fehler Checker untersucht den Code auf mögliche Laufzeitfehler.
Exceptions		ISO Standard with sophisticated exception handling	
Compileranweisungen	Verbose, Pedantisch, Wahl PIM/ISO Library, Linking Reihenfolge, Quadrupel Statistik, Indexgrenzen von Arrays prüfen, Preprocessor, etc.	68k / PPC Codegeneration, C / C++ Source Output, AltiVec (PPC), Float Optimization, Tests, etc. Siehe auch Pragmas.	unter anderem: __GEN_X86__, __GEN_C__, DIFADR16, FATFS, INDEX16, M2CMPSYM, NOOPTIMIZE, PROCINLINE, TARGET16, etc. Siehe auch Pragmas.

Bibliothek			
Bibliotheken / Schnittstellen	PIM Library komplett, ISO Library in Entwicklung	ISO Library, gewisse P1 spezifische Erweiterungen (Datentypen, MPW, 68k, etc.) und Macintosh Interface (Toolbox Universal Interface 3.4.2 und CarbonLib) als Foreign Definition Files.	ISO Library (ohne TermFile, LowLong und LowReal), PIM Library, eigene Erweiterungen, Win32 API, ANSI C Library
Debugging	Sämtliche Vorzüge des GDB (GNU Debugger)	Debugging auf Source Level, post mortem, interactive break point	Postmortem History mit Pragma aktivierbar.
Portabilität	Da GM2 auf vielen Zielplattformen läuft und auch Code für Windows generiert werden kann, erhöht er die Portabilität der RAMSES Software wesentlich.	P1 ist genau so auf die Mac OS Plattform zugeschnitten wie der bisherige MacMETH Compiler. Eine Verwendung dieses Compilers erhöht die Portabilität von RAMSES nicht, oder höchstens unwesentlich (da Ähnlichkeiten zwischen OS X und UNIX).	Erzeugt portablen C-Code. Bietet Schnittstellen zu Win32 API und ANSI C Libraries.
Zielplattformen	Linux/x86, SPARC, OS X (in Zukunft), Windows (CrossCompiler)	Mac OS und Mac OS X in MPW (v-7.3.6), Mac OS X native (ab 8.0).	Linux/x86, Windows. Generierter C-Code kann mit C-Compiler auf anderen Plattformen kompiliert werden (vorausgesetzt Libraries sind vorhanden).
Portierungsaufwand	Eine Anpassung der DM an GM2 sollte reichen. Schnittstellen für CarbonLib (OS X) müssten erstellt werden, minimaler Aufwand für Unterhalt UNIX/OS X Ports, da identischer Compiler. Win32 API vorhanden. Cross Compiler nach Windows vorhanden	Geringster Aufwand für Portierung nach OS X, da Schnittstellen für Toolbox und CarbonLib, zukünftig auch Cocoa vorhanden. Nachteil: separate Compiler für UNIX und Windows benötigt.	Aufwand in etwa gleich wie GM2. Dank intermediärem C Code evtl. etwas flexibler.

3.6 Benchmark Test von Programmen der Kompiler P1 und GM2

Auf Grund der Resultate der Kompiler Evaluation wurden die Kompiler P1 8.0 auf der Plattform Mac OS X/PCC und GM2 auf der Plattform x86/Linux installiert und ihrer Performanz mit Hilfe einfacher Benchmarkprogramme für Integer- (Sieve, Anhang B.1) und Gleitkommaoperationen (Ereal, Anhang B.2) getestet. Die Resultate dieser Tests sind in der Tabelle 3.6 aufgeführt. Sie ergänzen frühere Benchmark Tests, welche mit Modula-2 Kompilern durchgeführt wurden (Löffler et al., 1996). Augenfällig ist die um eine Grössenordnung bessere Performanz bezüglich Gleitkommaoperationen der Plattformen x86/Linux und PPC/Mac OS X gegenüber der Classic-Umgebung. Zu berücksichtigen ist dabei, dass der MacMETH Kompiler keinen nativen Code für die entsprechende Plattform generiert und demnach für die Ausführung des Codes eine virtuelle Plattform (Classic) benötigt, was die Ausführung des Codes zwangsläufig verlangsamt (Löffler et al., 1996).

Eingehendere Benchmark-Tests wurden mit dem P1-Kompiler auf verschiedenen Maschinen durchgeführt und mit den zur Zeit verwendeten Kompilern MacMETH und EPC verglichen. Diese Resultate sind im Kapitel 5.3 “Performanz des Codes” zu finden

Tabelle 5: Resultate der Benchmarktests mit den Programmen Sieve (Integeroperationen) und Ereal (Gleitkommaarithmetik) mit verschiedenen Kompilern auf verschiedenen Plattformen. Die für die Ausführung der Programme benötigte Zeit ist in Sekunden angegeben.

Kompiler	**Plattform**	**Prozessor**	**Sieve**	**Ereal**
P1 v.8	PPC/Mac OS X	G4 550 MHz	1.0	4.9
		G5 1.6 GHz	0.7	1.1
MacMETH	PPC/Classic	G4 550 MHz	2.5	332.8
GM2	x86/Linux	P3 800 MHz	1.6	2.4

4 Kompilationstests mit P1 Modula

Aufgrund der vorgängigen Evaluation verschiedener Kompiler hat sich P1 Modula als der zur Zeit vielversprechendste Kanditat für die Portierung der DM herausgestellt. Vor der Portierung der DM soll der Kompiler mit dem bestehenden DM Code getestet und die dabei auftretenden Fehler analysiert werden.

Als Ausgangslage für die Portierung dient der Code der BatchDM für den EPC Kompiler auf der Sun Plattform. Dieser Code umfasst die Schichten SysDep (Version Sun), DM (Version BatchDM), MW, AuxLib (AuxLibSE), ISIS und SciLib. Der Umfang dieser Schichten ist der Tabelle 6 zu entnehmen. Insbesondere ist zu beachten, dass die Reihenfolge, mit welcher die DEF-Module kompiliert werden aufgrund gegenseitiger Abhängigkeiten von entscheidender Bedeutung ist.

Tabelle 6: Umfang der getesteten Module. In der Tabelle ist die Anzahl aller Module pro Softwareschicht aufgeführt, welche für eine funktionsfähige RAMSES Software (BatchDM Version) benötigt werden.

Softwareschicht	**# DEF**	**# MOD**	**Total**
SysDep	5	3	8
DM (BatchDM)	36	35	71
MW	37	36	73
AuxLib (ohne AuxLibPD)	116	116	232
ISIS	21	21	42
SciLib	38	38	76
Total	**253**	**249**	**502**

4.1 Resultate der Testkompilation

Wie erwartet, gehen die grössten Probleme von Modulen der Schichten SysDep und DM aus. Während höhere Module nur von standardisierten Bibliotheken abhängen (PIM3) und ansonsten auf Schnittstellen tiefer liegender Module zugreifen, machen v.a. die Module aus SysDep exzessiven Gebrauch von Kompiler spezifischen Bibliotheken, z.B. um den Zugriff auf das Dateisystem zu ermöglichen.

Die Implementation solch tiefer Module muss im Falle einer Portierung der DM teils grundlegend geändert werden, Für Abhängigkeiten von kompilerspezifischen Bibliotheken muss Ersatz gefunden, oder aber eigene Algorithmen geschrieben werden. Diesem Aspekt muss bei einer Portierung die entsprechende Aufmerksamkeit geschenkt werden, für die Testevaluation des Kompilers ist er jedoch von untergeordneter Bedeutung.

An dieser Stelle von Interesse ist die Compilation von Modulen, bei welchen nur geringe Probleme zu erwarten wären, da sie lediglich auf standardisierten Sprachdefinitionen beruhen. Die Fehlerstatistik der Module der Schichten MW, AuxLib, ISIS und SciLib sind in der Tabelle 7 aufgeführt.

Die in der Tabelle 7 aufgeführten Fehler, können jeweils auf eine der drei folgenden Ursachen zurückgeführt werden: a) Fehlverhalten des Kompilers, b) Folgefehler und c) Code ist nicht ISO-kompatibel.

Für eine Portierung der Dialog Maschine, sind v.a. Fehler der Kategorie c) und beschränkt der Kategorie a) von Interesse. Folgefehler können ignoriert werden. Dank intensiver Zusammenarbeit mit Herr Wiedemann, dem Entwickler des P1 Kompilers, konnte das Verhalten des Kompilers wesentlich verbessert werden, so dass letztlich nur Fehler der Kategorie c) zur Behebung übrig bleiben, also die ISO-Kompatibilität der RAMSES Software.

Im Folgenden werden die aufgetretenen Fehler im Detail diskutiert. Erst werden die Kompilerfehler

Tabelle 7: Fehlertypen und deren Häufigkeit der Kompilation der Implemetationsmodule (.MOD) der von den Kompiler-Bibliotheken unabhängigen, abstrakten Ramses Softwareschichten. Die Fehlerstatistik wurde mit den Ergebnissen des P1-Kompilers v-8.0 erstellt. Gewisse Fehler sind Kompilerfehler und treten bei der Kompilation mit P1 v-8.4 nicht mehr auf. Diese Fehler sind in der Zeile *Comp* aufgeführt.

Schicht	**Fehlertypen**													**Warnungen**		
	n	**20**	**23**	**39**	**71**	**73**	**76**	**80**	**81**	**82**	**83**	**89**	**211**	**608**	**609**	**630**
MW		2			3	8		1	4	2					1	9
AuxLib	1	14	1	1		2	8	8	428	13	16	2	38	4	5	66
ISIS		17	1				1	1	1				2		7	8
SciLib		1						2	127		3		2		1	156
Total	**1**	**34**	**2**	**1**	**3**	**10**	**9**	**12**	**560**	**15**	**19**	**2**	**42**	**4**	**14**	**239**
Comp								7	7	2		2	3			

Fehlertypen *n:* no symfile found, *20:* identifier expected, *23:* ';' expected, *39:* ')' expected, *71:* identifier not exported from qualifying module, *73:* identifier not declared, *76:* this type is not expected, *80:* constant or constant expression expected, *81:* incompatible types, *82:* type incompatible operands, *83:* operation with incompatible type, *89:* range only with scalar types, *211:* type of function expected

Warnungen *608:* this pragmatype is not supported, *609:* different sizes of source and destination, *630:* variable probably not initialized

diskutiert, welche mit der neuesten Version 8.4 des P1 Kompilers nicht mehr auftreten und somit für künftige Portierungsbemühungen ausser Betracht fallen. Anschliessend werden für jede in der Tabelle 7 aufgeführte Fehlerkategorie Ursachen und mögliche Lösungswege aufgezeigt.

4.2 Kompilerfehler

Gewisse in der Tabelle 7 aufgeführte Fehler konnten als Kompilerfehler eruriert und dem P1-Entwickler A. Wiedemann mitgeteilt werden. Inwieweit diese Änderungen wirklich aufgrund von Abweichungen des ISO-Standards erfoderlich waren, oder jedoch als Ergänzung zum Standard in den P1 Kompiler integriert wurden, ist nicht vollends geklärt. Auf alle Fälle aber sind dies Fehlertypen, welche bei einer P1-Implementation der RAMSES Software nicht mehr berücksichtigt werden müssen und hier nur der Vollständigkeit halber aufgeführt werden.

80: constant or constant expression expected

Die Deklaration von Konstanten in Modulen führte zu Problemen, wenn der Konstanten ein Wert in Form einer Prozedur zugeordnert wurde, wie z.B. `CONST REALSize = INTEGER(SIZE(REAL));` in `IdentifyPars.MOD`. Dieser Fehler zog andere Fehler, z.B. `82: type incompatible operands` nach sich. Dieser Fehler sollte mit der Kompiler Version P1 v-8.4 höchstens als Folgefehler anderer Fehler bei der Konstantendeklaration auftreten.

81: incompatible types

Durch Zuweisungsinkompatibilität verursachte Fehler, deren Ursache unbekannt blieb:

```
File "../src/MOD/DFViewBase.MOD"; Line 389
#  410        ci := minVDType; (* don't declare undefined *)
#####             ^  81: incompatible types
 File "../src/MOD/DFViewBase.MOD"; Line 410
#  413            RadioButton( radBut2[ci], cl, lem, radButS2Txt[ci]); INC(cl);
```

```
#####                                  ^  81: incompatible types ^  81: incompatible types
 File "../src/MOD/DFViewBase.MOD"; Line 413
#  415            radBut2[ci] := notInstalledRadioButton;
#####                       ^  81: incompatible types
 File "../src/MOD/DFViewBase.MOD"; Line 415
 #  419          makeChoiceRadButSet2 := radBut2[curChoice2];
#####                                                  ^  81: incompatible types
 File "../src/MOD/DFViewBase.MOD"; Line 419
#  458      FOR i2 := minVDType TO maxChoices2 DO
#####                           ^  81: incompatible types
 File "../src/MOD/DFViewBase.MOD"; Line 458
#  459        ConvertValDefTypeToString(i2,radButS2Txt[i2]);
#####                                                ^  81: incompatible types
 File "../src/MOD/DFViewBase.MOD"; Line 459
#  477        forType  := WhichChoice2(minVDType, makeChoiceRadButSet2, radBut2);
#####                                          ^  81: incompatible types
 File "../src/MOD/DFViewBase.MOD"; Line 477
```

82: type incompatible operands

Diese Fehler wurden einerseits ausgelöst durch den direkten Vergleich von Listenelementen, ohne deren vorgängige Konvertierung in Ordinalzahlen, andererseits durch die Subtraktion zweier Ordinalzahlen (beide vom Typ `INTEGER`). Beiden Fehlern ging ein Fehler des Typs `80: constant or constant expression expected` voraus. Es ist anzunehmen, dass es sich dabei also lediglich um Folgefehler handelt.

89: range only with scalar types

Verantwortlich für diesen Fehlertyp ist die Aufspannung von Arrays mit Hilfe von Elementen aus einer Liste, ohne deren expliziten Konvertierung in Ordinalzahlen. Die Fehlermeldung ist insbesondere auch falsch, da es sich bei den verwendeten Indizes um Skalare handelt. Evtl. handelte es sich dabei auch um einen Folgefehler von `80: constant or constant expression expected`. Das Problem ist mit der Kompiler Version P1 v-8.4 behoben.

211: type of function expected

Dieser Fehler wurde durch `RETURN` Statements ausgelöst. Ursache dafür ist die Zuweisungsinkompatibilität vom deklarierten, zu dem tatsächlich zurückgegebenen `RETURN`-Wert. Es ist nicht klar, welche Zuweisungsinkompatibilität in folgenden Fälle vorlagen, mit dem neuen Kompiler v-8.4 aber nicht mehr auftreten:

```
#  484      IF t=notExistingTable THEN errcode := unknowntable; RETURN notExistingItem END(*IF*);
#####                                                                ^ 211: type of function expected
 File "../src/MOD/TableHandler.MOD"; Line 484
#  487        RETURN p
#####              ^ 211: type of function expected
 File "../src/MOD/TableHandler.MOD"; Line 487
#  490        RETURN notExistingItem (*. t .*)
#####              ^ 211: type of function expected
 File "../src/MOD/TableHandler.MOD"; Line 490
Modula-2 - Execution terminated!
```

Kompiler Exceptions

Insgesamt sind zwei Kompiler Exceptions aufgetreten, welche nicht in der Fehlerstatistik in Tabelle 7 aufgeführt und deren Ursachen unbekannt und ohne grösseres Interesse sind:

Selector.MOD:

```
{standard input}:unknown:Undefined local symbol L_P0_Selector$stub
```

SYAccBase.MOD:

```
unexpected exception in Kompiler: TRAP instruction
detected in line 496 at position 83
Modula-2 - Execution terminated!
{standard input}:unknown:Undefined local symbol L__M2_STACK$stub
{standard input}:unknown:Undefined local symbol L_RaiseError_1_SYBase$stub
{standard input}:unknown:Undefined local symbol L_AssignString_2_DMStrings$stub
{standard input}:unknown:Undefined local symbol L_Append_6_DMStrings$stub
{standard input}:unknown:Undefined local symbol L_GetErrMsg_1_SYDModBase$stub
{standard input}:unknown:Undefined local symbol L_DoNothing_11_SimBase$non_lazy_ptr
{standard input}:unknown:Undefined local symbol L_TraverseTree_21_TableHandler$stub
{standard input}:unknown:Undefined local symbol L_RStr_15_Errors$stub
{standard input}:unknown:Undefined local symbol L_SetInsert_10_Errors$stub
{standard input}:unknown:Undefined local symbol L_Str_12_Errors$stub
{standard input}:unknown:Undefined local symbol L_AppendInsert_11_Errors$stub
{standard input}:unknown:Undefined local symbol L_GetErrMsg_1_SYAccBase$non_lazy_ptr
{standard input}:unknown:Undefined local symbol L_Info_1_Errors$stub
```

4.3 Folgefehler

Die im Folgenden aufgeführten Fehlerkategorien sind direkte oder indirekte Folgefehler von Fehlern bezüglich der ISO-Kompatibilität des Codes und brauchen deshalb nicht berücksichtigt zu werden. Sie sind lediglich der Vollständigkeit halber aufgeführt.

Fehlerklasse 39: ')' expected

Dieser Fehler ist nur einmal aufgetreten in einer Konstanten Deklaration als Folgefehler von `23: ';' expected` (siehe ISO-Inkompatibilitäten). Es wird daher nicht näher auf diesen Fehlertyp eingegangen.

Fehlerklasse 71: identifier not exported from qualifying module

Diese Fehlermeldung erscheint, wenn ein Bezeichner aus einem Definitions Modul importiert werden soll, der Bezeichner aber nicht gefunden wird. Das kann grundsätslich zwei Ursachen haben. Entweder kann das zu importierende Modul nicht gefunden werden und der Fehler muss als Folgefehler von `no symfile found` angesehen werden. Andererseits ist es auch möglich, dass das zu importierende Modul zwar gefunden wird, dieses jedoch den Bezeichner tatsächlich nicht exportiert, oder Fehler bei dessen Kompilation aufgetreten sind.

Die drei Fehler dieses Typs sind auf einen Fehler des Typs `80: constant or constant expression expected` bei der Kompilation von `SystemMARK = MPt(0FFFFFFFEH);` im Definitions Modul MWTypes zurückzuführen.

Fehlerklasse 73: identifier not declared

Der Fehler taucht auf, wenn ein Bezeichner verwendet wird, der nicht deklariert und auch nicht aus einem Schnittstellenmodul importiert wurde. Bei den aufgetauchten Fehlern handelt es sich ausnahmslos

um Folgefehler des Fehlers `71: identifier not exported from qualifying module`.

Fehlerklasse 80: constant or constant expression expected

Für 10 von 12 Fehlern stellte sich das Problem als Kompilerfehler heraus und konnte mit einer Anpassung des Kompilers behoben werden. D.h., dass diese Fehler und alle resultierenden Folgefehler (`71: identifier not exported from qualifying module` wegen Fehler in Definitionsmodul) ignoriert werden können. Die zwei restlichen Fehler sind Folgefehler von `23: ';' expected` (siehe ISO-Inkompatibilitäten).

4.4 ISO-Inkompatibilitäten

Fehlerklasse n: no symfile found

Dieser Fehler kommt dann zustande, wenn ein zu importierendes Modul nicht gefunden werden kann. Diese Fehlermeldung kann verschiedene Ursachen haben.

- **Pfad zur gesuchten Symboldatei ist nicht im Suchpfad des Kompilers** — Die P1 Option `-symfile` *Pfad zu den Symboldateien* ermöglicht es, Verzeichnisse mit Symboldateien dem Kompiler mitzuteilen. Die Option kann mehrere male auf der Befehlszeile auftauchen. Das Verzeichnis des zu kompilierenden Moduls ist automatisch im Suchpfad des Kompilers.
- **Das entsprechende Definitionsmodul wurde noch nicht kompiliert** — Alle zu importierenden Definitionsmodule müssen kompiliert und als Symboldateien dem Kompiler bekannt sein. Es empfiehlt sich von daher zuerst alle Definitionsmodule und anschliessend alle Implementationsmodule zu kompilieren. Die Reihenfolge mit welcher die Definitionsmodule kompiliert werden ist kritisch und muss allenfalls angepasst werden. Zirkuläre Abhängigkeiten von Definitionsmodulen können nicht aufgelöst werden.
- **Die Symboldatei ist nicht in der Bibliothek des Kompilers vorhanden** — Das ist der Fall, wenn es sich nicht um ein eigenes Modul handelt, sondern um ein externes Modul, welches aus der Kompiler Bibliothek importiert werden soll, dort aber nicht vorhanden ist. Hauptursache für diesen Fehler sind Unterschiede zwischen der PIM- und der ISO-Standardbibliothek. Am einfachsten ist die Lösung dieses Problems, wenn es in der aktuellen Kompiler Bibliothek ein Pendant zu dem gesuchten Modul gibt. Ansonsten muss der betroffene Code neu implementiert werden, mit Hilfe von den in der Bibliothek zur Verfügung stehenden Module. Solche Änderungen am Code haben grossen Einfluss auf die Portabilität des Codes. Es ist deshalb wichtig, solche Änderungen nicht generell vorzunehmen, sondern innerhalb eines Kompiler Flags, welches nur auf der entsprechenden Plattform aktiviert wird. Diese Kategorie von Fehlern ist v.a. in den unteren Ramses Softwareschichten SysDep und BatchDM anzutreffen, welche durch die Bereitstellung eigener Schnittstellen, die höheren Schichten von den Kompiler spezifischen Bibliotheken abschirmen.

Fehlerklasse 20: identifier expected

Bei diesem Fehler erwartet der Kompiler im Quelltext einen Bezeichner (Variable, Konstante oder Prozedur). Alle Fehler dieses Typs haben entweder die Form `c = {a,b}` (Vergleich) oder `c := {a,b}` (Zuweisung), wobei `a` und `b` vom Typ `CARDINAL` und `c` vom Typ `BITSET` sind. ISO-Modula-2 verlangt in diesem Falle eine klare Deklaration des Typs. Obige Ausdrücke müssten demnach heissen: `c = BITSET{a,b}`, bzw. `c := BITSET{a,b}`. Diese Ausdrücke werden auch von anderen Sprachimplementationen wie z.B. PIM-Implementationen verstanden und schränken somit die Portabilität des Codes nicht ein. Sie können also als generelle Änderung des Codes übernommen werden.

Fehlerklasse 23: ';' expected

Das ist eine etwas irreführende Fehlermeldung, da sie a) nicht sagt, wo genau das Problem ist und b) in verschiedenen Situationen auftreten kann, für welche es nicht eine einheitliche Lösung gibt. So sind auch die beiden Fehler dieses Typs, welche in der Testkompilation der höheren Module aufgetaucht sind von anderer Natur und demnach anders zu lösen. Die einzige Gemeinsamkeit der Fehler ist, dass sie beide in einer Konstanten Deklaration auftreten und vom Fehler `80: constant or constant expression expected` gefolgt werden[17].

- **Konstantendefinition beinhaltet Variablen** — Eine Konstantendefinition welche Variablen beinhaltet wie z.B. `CONST blockSIZE = SIZE(addressingOnThisMachine^);` in ByteBlockIO.MOD, ist in ISO Modula-2 nicht zulässig[18]. Die Implementation muss der Situation entsprechend geändert werden, was gute Kentnisse der Software abverlangt. In obigem Fall konnte das Problem folgendermassen gelöst werden: `CONST blockSIZE = LOCSPERBYTE;`. Natürlich können solche Änderungen die Qualität des Codes (in diesem Falle die Flexibilität) beeinträchtigen und sollten auf alle Fälle nicht generell, sondern Kompiler spezifisch, d.h. innerhalb eines Kompiler Flags vorgenommen werden.

- **Typ `BITSET` nicht deklariert** — Das Problem ist hier dasselbe wie bei Fehler `20: identifier expected`, ausser dass die fehlende `BITSET`-Deklaration nicht in einem Ausdruck, sondern in einer Konstanten-Definition auftritt. Zu beheben ist dieser Fehler auf dieselbe generelle Art durch z.B. `CONST neverDeclared = BITSET{};` anstelle von `CONST neverDeclared = {};`.

Fehlerklasse 81: incompatible types

Dieser Fehler tritt dann auf, wenn nach ISO-Standard verschiedene Typen einander zugewiesen oder miteinander verglichen werden. In der Sprachdefinition PIM-123 gelten folgende Zuweisungs- Kompatibilitäten: (`INTEGER`, `CARDINAL`, `LONGINT`, `LONGCARD`) und (`REAL`, `LONGREAL`), sowie (`BYTE`, alle Datentypen der Grösse 1 Byte) und (`WORD`, alle Datentypen der Grösse 2 Byte) (Wirth et al., 1992). `ARRAY OF BYTE` ist mit jedem Datentyp kompatibel und Variablen vom Typ `ADDRESS = POINTER TO BYTE` sind kompatibel mit jedem Pointertyp und können für Adressarithmetik mit Integeroperationen verwendet werden.

Der ISO-Standard ist weit restriktiver. Zwar können mit INTEGER und CARDINAL Typen untereinander zugewiesen werden. Alle anderen Zuweisungen gelten aber als nicht kompatibel. Je nach Art von Zuweisung müssen andere Lösungen gesucht werden. Man könnte versucht sein, gewisse harmlose Typen von Zuweisungen mit Hilfe eines CASTS (unsafe type transfer) zu erzwingen, wenn die beiden Datentypen dieselbe Grösse haben. Andere Konvertierungen, wo sich die Datentypen von der Grösse her unterscheiden, erfordern einen überwachten Typtransfer (safe type transfer) .

- **REAL/LONGREAL Inkompatibilität** — Der ISO-Standard erlaubt keine Zuweisungskompatibilität von Variablen des Typs `REAL` nach `LONGREAL` und umgekehrt. Eine explizite Zuweisung in Form eines Castings (z.B. `r := REAL(lr);`) ist falsch, da eine solche unüberwachte Konvertierung aufgrund unterschiedlicher Grösse der Datentypen zu unerwarteten Resultaten führen kann.

  Dahingegen stellt der ISO Standard die Standardfunktionen `FLOAT(): REAL;` und `LFLOAT(): LONGREAL` zur überwachten Konvertierung der Typen `REAL` bzw. `LONGREAL` zur Verfügung. Die

[17] Die Fehlermeldung `80: constant or constant expression expected` stellte sich als Kompilerfehler heraus und ist mittlerweile behoben.

[18] Zusätzlich wurden bei diesem Ausdruck die Fehler `39: ')' expected` und `80: constant or constant expression expected` gemeldet

Verwendung dieser Prozeduren erfordert jedoch aufgrund der Portabilität eine Unterscheidung der Plattform mit Hilfe konditionaler Kompilation mit Kompiler Flags, da PIM Kompiler nur die Standardfunktion `FLOAT(): REAL;` auf sicher unterstützen.

Für die Implementation der erforderlichen Konvertierungs-Funktionalität, empfiehlt sich aufgrund der Schlankheit des Codes und dessen Portabilität die Erstellung eigener, plattformunabhängiger Konvertierungsfunktionen in einer tiefen Softwareschicht, z.B. `SR(lr: LONGREAL): REAL;` bzw. `LR(sr: REAL): LONGREAL` in `DMPortab`, welche je nach Kompiler auf unterschiedliche Weise implementiert sein können.

- **ADDRESS/LONGCARD Inkompatibilität** — Der ISO Standard erlaubt keine implizite Zuweisung von Variablen des opaken Typs `ADDRESS` zu Variablen des Typs `LONGCARD` und umgekehrt. Die Zuweisung muss explizit gemacht werden, d.h. mit einer expliziten Transformation (casting) vom einen zum anderen Typ, z.B. durch `lc := LONGCARD(adr);` oder `adr := ADDRESS(lc);`

  Eine weitere Möglichkeit zur Umgehung dieses Problems läge in der Verwendung der für die Adressarithmetik vorgesehenen ISO-Prozeduren[19] aus dem Pseudomodul `SYSTEM`. Diese Vorgehen wäre vorteilhaft bezüglich Eleganz, nachteilig wäre die ISO-Spezifizität dieser Lösung.

  Diese Fehlerklasse steht in engem Zusammenhang mit Fehlern des Typs `83: operation with incompatible type`, die auftreten, wenn Adressarithmetik mit gewöhnlichen Operatoren für die Integerarithmetik durchgeführt werden. Mit den oben genannten Lösungen wird auch dieses Problem umgangen.

- **BYTE/CHAR Inkompatibilität** — Da die Grösse dieser beiden Typen auf allen fraglichen Plattformen identisch ist, kann eine erzwungene Typen-Konvertierung ohne unerwartete Konsequenzen erfolgen, z.B. `b := BYTE(ch);`. Eine solche Änderung kann generell gemacht werden und schränkt die Portabilität des Codes nicht wesentlich ein.

- **BYTE Zuweisung** — Die direkte Zuweisung von Bitmustern zu einer Variable vom Typ `BYTE` benötigt in ISO Modula-2 immer eine erzwungene Typumwandlung, z.B. `b := BYTE(001C)`, was ebenfalls ohne negative Konsequenzen für die Portabilität des Codes bleibt.

- **WORD/BITSET Inkompatibilität** — Das Problem ist identisch mit dem Kompatibilitäts-Problem `BYTE/CHAR`, ausser dass es sich hier um Datentypen der Grösse 2 Byte handelt. Eine explizite Konvertierung kann ohne Risiko und ohne Einschränkung der Portabilität vorgenommen werden.

Eine genaue Quantifizierung der verschiedenen Fehlertypen innerhalb dieser Klasse kann nur manuell erfolgen. Dies wurde nicht gemacht. Der mit Abstand grösste Anteil der Fehler ist jedoch der Inkompatibilität `REAL/LONGREAL` und der `BYTE`-Zuweisung zuzuschreiben.

Fehlerklasse 82: type incompatible operands

Dieser Fehler tritt auf, wenn Operationen auf unzulässige Typen angewendet werden. Der ISO-Standard erlaubt keine gemischten Ausdrücke also wie z.B. Integeroperationen mit Variablen vom Typ `ADDRESS`. Weitere Fehler dieses Typs tauchten auf mit dem Vergleich, bzw. Operationen zwischen ganzzahligen Variablen und Konstanten vom Typ `CARDINAL` oder `LONGCARD` und `INTEGER` oder `LONGINT`. Gewisse Fehler können auch als Fogelfehler von `80: constant or constant expression expected` betrachtet werden.

[19] Die ISO-Version des Pseudomoduls `SYSTEM` bietet für die Adressarithmetik folgende Prozeduren an: `ADDADR(addr: ADDRESS; offset: CARDINAL): ADDRESS;`, `SUBADR(addr: ADDRESS; offset: CARDINAL): ADDRESS;`, `DIFADR(addr1, addr2: ADDRESS): INTEGER;` und `MAKEADR(val: zz-type): ADDRESS;`

Fehlerklasse 83: operation with incompatible type

Dieser Fehlertyp ist sehr eng mit dem Fehler `82: type incompatible operands` verwandt. Alle aufgetretenen Fehler dieses Typs beziehen sich jedoch auf die unterschiedliche Adressarithmetik in PIM- und ISO-Modula.

In ISO-Modula ist es nicht zulässig Integer-Operationen mit Variablen des Typs `ADDRESS` durchzuführen. Eine Möglichkeit dieses Problem zu umgehen besteht darin, entweder die Variablen des Typs `ADDRESS` vorgängig in `LONGCARD` Variablen zu transformieren, oder aber die von der ISO-Version des Pseudomoduls `SYSTEM` angebotenen Prozeduren für die Adressarithmetik zu verwenden. Diese Möglichkeiten wurden schon unter dem Aspekt der Zuweisungskompatibilität (`81: incompatible types`) diskutiert.

Fehlerklasse 211: type of function expected

Dieser Fehler entspricht dem Fehler `81: incompatible types` im Falle, wo von einer Prozedur ein `RETURN`-Wert zurückgegeben wird, der mit dem deklarierten `RETURN`-Wert nicht Zuweisungskompatibel ist.

Zusammenfassend können folgende Fehlerursachen als Hauptprobleme allfälliger Portierungsbestrebungen betrachtet werden:

- Importe nicht vorhandener Bibliotheksmodule (PIM- vs. ISO-Standardbibliotheken). V.a. ein Problem systemnaher plattformabhängiger Softwareschichten, welches kompilerspezifisch den Import äquivalenter Module, oder eine Neuimplementation der betroffenen Prozeduren erfordert.
- `BITSET`-Deklaration von Ausdrücken der Art `c := {a,b}`, die gemäss ISO als `c := BITSET{a,b}` zu schreiben sind. Das Problem tritt nur sporadisch auf. Es ist einfach und generisch für alle zu unterstützenden Kompiler zu beheben.
- Die ISO-Zuweisungsinkompatibilitäten gewisser Typen (v.a. REAL/LONGREAL und BYTE-Zuweisungen) können teils mit einem CAST (unsafe type transfer) umgangen werden (z.B. für "sichere" BYTE-Zuweisungen wie `b := BYTE(001C)`). In der Regel wird jedoch von einem unsafe type transfer abgeraten. Besser ist die Verwendung von Prozeduren der ISO-Library für den Typtransfer (safe type transfer), was jedoch Auswirkungen auf die Portabilität des Codes haben kann. Die meisten solchen Probleme dieser Art sind deswegen aufwändig und kompilerspezifisch von Hand zu lösen.
- Probleme bei der Anwendung von Operationen, Relationen und Prozeduren auf nicht-kompatible Typen, tritt v.a. bei der Adressarithmetik auf und erfordert entweder eine Typumwandlung, oder die Verwendung von ISO-Bibliotheken. Das Problem entspricht weitgehend jenem der Zuweisungsinkompatibilitäten unter ISO Modula-2

5 Test-Implementation der DM mit P1 Modula

Aufgrund der vorgängigen Analysen wird P1-Modula als der zur Zeit vielversprechendste Kandidat für eine Portierung der DM angesehen. Viele der während der Kompilertests aufgetreten Fehler können relativ einfach auf eine generische Art, d.h. ohne die Funktionalität des Codes bezüglich bestehenden Plattformen einzuschränken, behoben werden. Im Gegenteil kann der Codebestand bezüglich Portabilität von einer Portierung auf den Kompiler P1 profitieren, da dies auch zukünftige Portierungen auf ISO-Kompiler, insbesondere auf Unix/Linux-Plattformen wesentlich erleichtern sollte.

Die im folgenden diskutierte Portierung der DM umfasst lediglich Module welche für die Batch-Verarbeitung von Modellen benötigt werden. Insbesondere fallen dadurch sämtliche Module weg, die für einen interaktiven Betrieb der RAMSES-Software benötigt werden, inkl. grafische Ausgabe der Resultate.

Folgende generelle Vorgehensweisen wurde für die Portierung der Dialog Machine gewählt:

1. Wo immer möglich, soll der Code generisch gehalten werden, so dass er auf allen Plattformen kompilierbar bleibt. Das gilt v.a. für die höheren Softwareschichten AuxLib, MW, ISIS und SciLib.

2. Wo es nicht möglich ist, Änderungen generisch zu gestalten, nämlich dort wo ISO spezifische Bibliotheken benötigt werden, wird ein Kompiler-Flag eingeführt. ISO spezifische Anpassungen an die Sprachdefinition sollten aufgrund der mit den Tests gewonnenen Erkenntnissen nicht nötig sein. Anpassungen an die Sprache (z.B. wegen Problemen der Typeninkompatibilität) können i.d.R.[20] auf generische Art gelöst werden.

3. Wo keine ISO-Bibliotheksmodule für die Implementation zur Verfügung stehen, müssen Plattformspezifische Bibliotheken verwendet werden. Für Low-Level Funktionalitäten soll auf Unix spezifische C-Schnittstellen zugegriffen werden. Für höhere Funktionalitäten, wie z.B. das grafische I/O sollen zu einem späteren Zeitpunkt plattformspezifische Bibliotheken (Carbon unter OSX) verwendet werden. Solche plattformspezifischen Anpassungen dürfen lediglich in den tieferen Softwareschichten SysDep und BatchDM vorgenommen werden.

5.1 Implementationsaufwand

Low-Level Funktionalität: SysDep

Die Portierung von SysDep erfolgt ausgehend von der Implementation für die EPC-Version, welche für die BatchDM auf der Sun existiert. Der von Anfang an auf UNIX Plattformen entwickelte EPC-Kompiler stellt eine Menge an nicht-standartisierten UNIX spezifischen Bibliotheken zur Verfügung, welche für die Implementation unter P1 nicht zur Verfügung stehen. Die ISO-Bibliothek stellt nebst einfachstem File-I/O kaum Module zur Interaktion mit dem Betriebssystem zur Verfügung. D.h. dass die Implementation von SysDep unter exzessiver Verwendung von C-Schnittstellen grundlegend geändert werden muss, was zwar einen grossen Aufwand, jedoch auch gewisse Vorteile, was die Portabilität des Codes betrifft, mit sich bringt.

Folgende Funktionalitäten müssen über C-Schnittstellen zur Verfügung gestellt werden:

- Verwaltung des Dateisystems, Datei-I/O
- Speicherverwaltung
- Zugriff auf Systemzeit und Zeitkonvertierung

[20] Eine Ausnahme ist die Verwendung der ISO Standardprozedur LFOAT in Portab.MOD

- Zugriff auf Umgebungsvariablen
- Ausführung externer Programme

High-Level Module

Die Portierung der höheren Module der RAMSES-Schichten DM, ModelWorks, AuxLib, ISIS und SciLib, sowie darauf aufbauender Modelle und Tests umfasst in erster Linie eine syntaktische Portierung des Codes auf die ISO-Sprachdefinition. Dabei sind die Fehler aus der Tabelle 10 zu berücksichtigen. Viele dieser Fehler können relativ einfach behoben werden, erfordern trotzdem eine Menge Handarbeit, denn eine automatische Portierung des Codes ist nicht möglich.

5.2 Das Modul SysDep

Die Portierung der SysDep Softwareschicht erforderte eine Neuimplementation des Moduls SysDep.mod (Anhang C.2), mit Anlehung an die Implementation für den EPC Kompiler. Zusätzlich musste das Modul CCalls.def (Anhang C.1) erstellt werden, welches die Schnittstelle zu den C-Bibliotheken des Systems darstellt.

Im Modul SysDep.mod wurden die plattformabhängigen Implementationen in folgende innere Module gekapselt, um die Portierung auf weiter Plattformen zu vereinfachen.

- `UnixStdio` für Datei-I/O
- `UnixFiles` für Dateimanagement
- `UnixDirectory` für Verzeichnismanagement
- `Files` Repräsentation von Dateien
- `Directory` Repräsentation von Verzeichnissen
- `Clock` Abfrage und Konvertierung der Systemzeit
- `UnixParam` für die Verarbeitung von Argumenten der Befehlszeile

Der grosse Aufwand für die Neuimplementation des SysDep-Moduls bezahlt sich in gewisser Weise auch aus. Mit der Portierung erhält man eine SysDep Version, welche lediglich von standardisierten ISO-Bibliotheken und einer standardisierten Schnittstelle zu C abhängig ist. Da alle bekannten Modula2-Kompiler auf UNIX Plattformen C-Schnittstellen in einer ähnlichen Weise unterstützen, sollte das resultierende SysDep mit einem sehr geringen Aufwand auf alle ISO-Kompiler auf UNIX Plattformen portiert werden können.

Um die SysDep-Implementation zu testen, wurde das Programm TestSysDep.MOD (Anhang D.2) geschrieben. Es testet alle in der Schnittstelle SysDep.def veröffentlichten Prozeduren und überprüft die Implementation auf korrektes Verhalten. Das Programm kann demnach für die Überprüfung aller SysDep-Implementationen verwendet werden.

5.3 Performanz des Codes

Mit dem Benchmark-Programm Benchmark.MOD (Anhang B.3) wurden die DM-Implementationen auf den Plattformen EPC/Sun, MacMETH/Classic und P1/PPC auf ihre Performanz getestet. Zu beachten ist, dass die Benchmark-Tests nicht nur die Performanz des vom jeweiligen Kompiler erzeugten Codes testet, sondern gleichzeitig jene der jeweiligen DM-Implementation.

Die Tabelle 8 zeigt, dass die Gesamt-Performanz der P1-DM (grün hinterlegt) jene der EPC-Implementation bei vergleichbarer Taktrate um Faktor 2 - 3 übersteigt. Dabei muss jedoch berücksichtig werden, dass die Fliesskommaarithmetik (REAL arithemtic + * /), welche für die RAMSES Simulationsumgebung von zentraler Bedeutung ist auf der Sun Plattform schneller ist als unter der P1-DM. Dieser Mangel in der Gleitkommaarithmetik auf der PPC-Plattform wird jedoch durch die neuen G5 Prozessoren bei Weitem wett gemacht. Unübertroffen ist die Prformanz von P1 bei transzendentalen Berechnungen, was auf die Leistungen der AltiVec auf der PPC Plattform zurückzuführen ist.

Die Performanz der P1-DM Implementation übertrifft die MacMETH/Classic Implementation um Faktor 4 - 5, wobei berücksichtigt werden muss, dass Classic unter Mac OS X lediglich emuliert wird und von daher zwangsläufig an Performanz einbüsst. Auf PPC-Rechnern neuerer Generation steht jedoch die Option, Mac OS 9 direkt zu booten gar nicht mehr zur Option.

5.4 Qualität und Vergleichbarkeit des Codes

Für die Simulation von Modellen, ist es erforderlich, dass die Resultate, welche auf verschiedenen Plattformen mit denselben Modellen erzielt werden miteinander vergleichbar, von Vorteil identisch sind. Das wird bisher mit den DM-Versionen auf den Plattformen MacMETH/Mac und EPC/Sun erreicht. Mit der P1 BatchDM-Version sind diese Resultate leider nicht reproduzierbar. Dies ist auf die Fliesskomma-Arithmetik des P1-Kompilers zurückzuführen, welcher sämtliche Ausdrücke vom Typ REAL (32-bit) in double-precision (64-bit) berechnet, und erst das Resultat des Ausdruckes in einen single-precision Wert umwandelt. Ein ähnliches Verhalten weist auch der Macintosh KKompilerompiler Compile20 auf, dessen Resultate durchaus mit den Resultaten des P1-Kompilers vergleichbar sind. Der SymDiff-Vergleich des Ouptuts der Testsuite, kompiliert mit den beiden Kompilern P1 und Compile20 ist in der Tabelle 9 aufgeführt.

Insgesamt konnten 17 der 40 Tests nicht miteinander verglichen werden. 3 Tests konnten aufgrund einer zur Zeit noch unbekannten Fehlerursachen der Fliesskomma-Arithmetik (`program terminated due to exception: real number invalid operation`) nicht durchgeführt werden, was im Falle von TstDMConv 4 Folgefehler mit sich zieht. TstDMFloatEnv wurde mit der Fehlermeldung `Exceptions at start up` beendet. Die 6 ISIS-Tests (Namen enden mit grossem S) wurden wegen dem Fehler `FATAL - not yet implemented: RetrieveResource in DMResources` frühzeitig abgebrochen. 4 Tests existieren nicht.

Die 22 miteinander verglichenen Tests weisen unterschiedliche Resultate auf. Bei 12 Tests konnte Identität zwischen den P1 und den Compile20 Resultaten festgestellt werden. 3 Tests wiesen insofern nur moderate Unterschiede auf, dass Sie sich lediglich bezüglich Rt (reelle Zahlen) unterscheiden, Rz (reelle Zahlen mit Differenz ungleich 0) jedoch identisch und der Fehler somit sehr gering ist. Die restlichen 7 Tests weisen z.T. grössere bis sehr grosse Abweichungen vom Compile20 Referenzoutput auf und bedürften deswegen einer genaueren Analyse. Eine mögliche und aufgrund der zahlreichen identischen Resultaten naheliegende Fehlerursache besteht in einer fehlerhaften Vorgehensweise, z.B. fehlerhafte Kompilation, falscher Referenzoutput, etc.

Aus dem Vergleich der Testprogramme mit dem Referenzoutput von Compile20 resultiert die Fehlerstatistik in Tabelle 10.

Tabelle 8: Performance von Benchmark.MOD (Appendix B.3) in den Implementationen MacMETH, EPC-Modula und P1auf verschiedenen Maschinen.

Macintosh machines under OS X 10.3.7 using GenBench (MacMETH 3.2.8/RAMSES 3.1.7 or 3.1.8) (b) and Benchmark.MOD in batch mode computed (testTime 60, scale 50 adjusted for 100 sec test time)
All settings default of RAMSES 3.1.7 (alwaysSANE on, DMMathLib (SANE), Compile)
P1 Modula-2 code default compilation (using RMSP1 v0.93)
Sun machines under various Solaris systems using always epc Modula-2 2.0.9.5 compiler and BatchDM/RASS 3.1.7 or 3.1.8
Standard code generation without optimization (typical use)
Note on the Sun it is not possible to control checks from within the code, thus tests g and h, plus i and j, respectively, show same performance
Test time = 100 sec (for explanation see Wirth (1981) and Loeffler et al. (1995)). All mesurements made while machines were spared from extra computational loads besides the benchmark program.
Since all measurements were made under realistic conditions, results are not necessarily reproducible, since the host machines might have been affected e.g. by network activities or other system processes.

Measurements

	(a) Loop size	test Time 100	(b): Lo29,Tab.3.1	(b)/(a)	inawa	megan	atitlan	malawi	biwa/eerie	huron	grandeur	boann
Machine		Lilith	Quadra 700 68040 25 MHz		SPARC Ultra-1 (143, 167, 200 MHz?)	Cube G4 450 MHz	SPARC Ultra-250 (300-400 MHz?)	SPARC Ultra-30 (250-300 MHz?)	SPARC Ultra-10 (300-440 MHz?)	Blade 100 500 MHz UltraSPARC-Iie	PowerBook G4 17" 1 GHz	iMac G4 17" 1 Ghz
a: Empty REPEAT	20000	535	272,747,475	13,637	74,888.9	149,555.6	81,333.3	90,611.1	134,888.9	153,583.3	318,694.4	302,722.2
b: Empty WHILE	20000	557	271,840,000	13,592	55,083.3	136,166.7	82,361.1	85,500.0	127,527.8	145,305.6	287,194.4	270,583.3
c: Empty FOR	20000	703	223,020,000	11,151	75,027.8	128,444.4	122,611.1	131,777.8	196,555.6	223,888.9	241,666.7	229,444.4
d: CARDINAL arithmetic * DIV	10000	312	40,623,762	4,062	11,750.0	53,277.8	24,888.9	19,444.4	30,361.1	43,666.7	153,972.2	158,222.2
e: REAL arithmetic + * /	5000	217	1,790,000	358	121,666.7	3,722.2	176,611.1	192,638.9	287,027.8	326,944.4	4,833.3	6,027.8
f: REAL Sin Exp Ln Sqrt	500	145	75,000	150	23,750.0	9,277.8	38,750.0	38,916.7	57,555.6	75,194.4	15,138.9	16,777.8
g: CARDINAL vector by element assignment	20000	182	52,950,495	2,648	19,722.2	22,194.4	24,555.6	28,555.6	42,666.7	48,638.9	86,055.6	94,194.4
h: like g but with index range checks	20000	148	30,772,277	1,539	19,750.0	30,222.2	25,694.4	28,583.3	42,666.7	48,638.9	66,250.0	68,972.2
i: CARDINAL matrix by element assignment	1	328	3,859	3,859	37,277.8	112,750.0	61,777.8	65,083.3	96,222.2	109,638.9	238,305.6	244,166.7
j: like i but with index range checks	1	273	2,605	2,605	37,277.8	61,833.3	59,333.3	65,111.1	96,277.8	109,666.7	134,555.6	139,638.9
k: Calling empty PROC procedure	20000	240	82,020,000	4,101	37,666.7	56,777.8	41,166.7	44,916.7	67,138.9	76,888.9	86,333.3	88,444.4
l: like k but 4 parameter PROCEDURE	20000	157	66,100,000	3,305	16,277.8	46,666.7	22,444.4	23,916.7	35,111.1	40,027.8	74,055.6	77,222.2
m: CARDINAL vector assignment (block move)	500	105	1,145,000	2,290	27,972.2	36,361.1	44,000.0	48,916.7	73,055.6	83,666.7	69,333.3	79,638.9
n: Linear list traversal	500	208	2,192,500	4,385	21,611.1	66,416.7	33,500.0	38,000.0	56,722.2	64,527.8	138,944.4	146,472.2
o: Read from file	5000	345	378,788	76	5,111.1	20,611.1	7,777.8	8,777.8	12,388.9	13,333.3	30,500.0	31,305.6
Average performance		**297.0**		**4,517.2**	**38,988.9**	**62,285.2**	**56,453.7**	**60,716.7**	**90,411.1**	**104,240.7**	**129,722.2**	**130,255.6**
p: like e but LONGREAL (basic arithmetic)					91,500.00	3,333.33	137,750.00	148,194.44	221,083.33	252,000.00	5,250.00	5,361.11
q: like f but LONGREAL (transcendental fcts.)					24,666.67	7,305.56	44,916.67	39,861.11	61,694.44	79,805.56	12,194.44	12,722.22

Performance Normalized

a: Empty REPEAT	1	25.5	140.0	279.5	152.0	169.4	252.1	287.1	595.7	565.8
b: Empty WHILE	1	24.4	99.0	244.6	148.0	153.6	229.1	261.0	515.9	486.1
c: Empty FOR	1	15.9	106.7	182.6	174.3	187.4	279.5	318.3	343.6	326.2
d: CARDINAL arithmetic * DIV	1	13.0	37.7	170.9	79.9	62.4	97.4	140.1	494.0	507.7
e: REAL arithmetic + * /	1	1.7	561.5	17.2	815.1	889.1	1,324.7	1,509.0	22.3	27.8
f: REAL Sin Exp Ln Sqrt	1	1.0	163.8	64.0	267.2	268.4	396.9	518.6	104.4	115.7
g: CARDINAL vector by element assignment	1	14.6	108.6	122.2	135.2	157.2	234.9	267.7	473.7	518.5
h: like g but with index range checks	1	10.4	133.1	203.7	173.2	192.7	287.6	327.9	446.6	465.0
i: CARDINAL matrix by element assignment	1	11.8	113.5	343.4	188.2	198.2	293.1	333.9	725.8	743.7
j: like i but with index range checks	1	9.5	136.4	226.2	217.1	238.2	352.2	401.2	492.3	510.9
k: Calling empty PROC procedure	1	17.1	156.9	236.6	171.5	187.2	279.7	320.4	359.7	368.5
l: like k but 4 parameter PROCEDURE	1	21.1	103.9	297.9	143.3	152.7	224.1	255.5	472.7	492.9
m: CARDINAL vector assignment (block move)	1	21.8	266.4	346.3	419.0	465.9	695.8	796.8	660.3	758.5
n: Linear list traversal	1	21.0	103.7	318.8	160.8	182.4	272.3	309.7	666.9	703.1
o: Read from file	1	0.2	14.8	59.7	22.5	25.4	35.9	38.6	88.4	90.7
Average performance	**1.0**	**13.9**	**149.7**	**207.6**	**217.8**	**235.3**	**350.4**	**405.7**	**430.8**	**445.4**
p: like e but LONGREAL (basic arithmetic)			27.450	1.000	41.325	44.458	66.325	75.600	1.575	1.608
q: like f but LONGREAL (transcendental fcts.)			3.376	1.000	6.148	5.456	8.445	10.924	1.669	1.741
p/e			0.752	0.896	0.780	0.769	0.770	0.771	1.086	0.889
q/f			1.039	0.787	1.159	1.024	1.072	1.061	0.806	0.758

Measurements

Machine	(a) Loop size	borea Cube G4 1.2 GHz	ito PowerBook PowerBook G4 15" 1.33 GHz (FPU)	ito PowerBook PowerBook G4 15" 1.33 GHz (All SANE)	Sophie's G5 G5 1.6 GHz	Oliver Gardi's PowerBook PowerBook G4 15" 550 MHz P1	megan Cube G4 450 MHz	grandeur PowerBook G4 17" 1 GHz	boann iMac G4 17" 1 Ghz	borea Cube G4 1.2 GHz	Sophie's G5 G5 1.6 GHz
a: Empty REPEAT	20000	371,027.8	412,555.6	409,916.7	211,583.3	614,861.1	810,138.9	1,258,833.3	1,139,500.0	1,318,555.6	1,562,277.8
b: Empty WHILE	20000	324,555.6	374,861.1	374,611.1	234,000.0	812,138.9	892,138.9	1,653,722.2	1,705,083.3	1,850,611.1	2,033,555.6
c: Empty FOR	20000	274,166.7	311,805.6	315,833.3	299,388.9	621,027.8	906,361.1	1,184,000.0	1,260,722.2	1,234,250.0	2,097,666.7
d: CARDINAL arithmetic * DIV	10000	182,638.9	207,805.6	207,583.3	128,138.9	195,027.8	178,388.9	330,944.4	362,694.4	380,333.3	374,750.0
e: REAL arithmetic + * /	5000	6,583.3	7,305.6	7,694.4	5,527.8	237,916.7	234,527.8	408,555.6	446,194.4	515,416.7	1,004,583.3
f: REAL Sin Exp Ln Sqrt	500	19,500.0	20,916.7	21,388.9	14,638.9	126,027.8	150,222.2	210,500.0	231,166.7	270,527.8	484,361.1
g: CARDINAL vector by element assignment	20000	108,583.3	133,277.8	133,583.3	71,750.0	201,472.2	193,944.4	342,666.7	376,333.3	435,694.4	350,972.2
h: like g but with index range checks	20000	78,944.4	91,888.9	91,361.1	65,861.1	132,805.6	70,555.6	222,305.6	232,472.2	271,888.9	7,527.8
i: CARDINAL matrix by element assignment	1	271,083.3	322,138.9	331,277.8	258,861.1	547,361.1	402,333.3	1,069,055.6	1,209,222.2	1,366,777.8	1,315,416.7
j: like i but with index range checks	1	156,333.3	179,888.9	181,555.6	153,361.1	246,694.4	138,750.0	423,694.4	461,916.7	535,222.2	471,944.4
k: Calling empty PROC procedure	20000	101,805.6	98,972.2	112,277.8	134,111.1	175,972.2	178,388.9	319,166.7	325,500.0	379,222.2	661,138.9
l: like k but 4 parameter PROCEDURE	20000	87,611.1	86,000.0	98,777.8	107,666.7	155,777.8	153,666.7	276,416.7	287,805.6	335,194.4	689,416.7
m: CARDINAL vector assignment (block move)	500	90,555.6	93,916.7	94,055.6	116,944.4	88,666.7	110,222.2	158,416.7	162,305.6	190,000.0	251,000.0
n: Linear list traversal	500	164,416.7	187,916.7	188,916.7	52,694.4	294,555.6	396,250.0	554,388.9	560,277.8	642,444.4	795,000.0
o: Read from file	5000	36,611.1	34,055.6	40,722.2	37,972.2	37,777.8	28,722.2	67,805.6	68,305.6	79,222.2	102,083.3
Average performance		**151,627.8**	**170,887.0**	**173,970.4**	**126,166.7**	**299,205.6**	**322,974.1**	**565,364.8**	**588,633.3**	**653,690.7**	**813,446.3**
p: like e but LONGREAL (basic arithmetic)		6,083.33			5,305.56	237,805.56	233,527.78	442,250.00	445,805.6	515,194.44	858,657.4
q: like f but LONGREAL (transcendental fcts.)		14,777.78			11,583.33	126,333.33	150,000.00	228,166.67	232,527.8	270,916.67	451,527.8

Performance Normalized

a: Empty REPEAT	693.5	771.1	766.2	395.5	1,149.3	1,514.3	2,353.0	2,129.9	2,464.6	2,920.1
b: Empty WHILE	583.0	673.4	673.0	420.4	1,458.9	1,602.6	2,970.8	3,063.0	3,324.5	3,653.1
c: Empty FOR	389.8	443.3	449.1	425.7	883.0	1,288.7	1,683.4	1,792.5	1,754.9	2,982.5
d: CARDINAL arithmetic * DIV	586.0	666.8	666.0	411.1	625.8	572.4	1,061.9	1,163.7	1,220.3	1,202.4
e: REAL arithmetic + * /	30.4	33.7	35.5	25.5	1,098.1	1,082.4	1,885.6	2,059.4	2,378.8	4,636.5
f: REAL Sin Exp Ln Sqrt	134.5	144.3	147.5	101.0	869.2	1,036.0	1,451.7	1,594.3	1,865.7	3,340.4
g: CARDINAL vector by element assignment	597.7	733.6	735.3	395.0	1,109.0	1,067.6	1,886.2	2,071.6	2,398.3	1,932.0
h: like g but with index range checks	532.2	619.5	615.9	444.0	895.3	475.7	1,498.7	1,567.2	1,833.0	50.7
i: CARDINAL matrix by element assignment	825.6	981.1	1,009.0	788.4	1,667.1	1,225.4	3,256.0	3,682.9	4,162.8	4,006.3
j: like i but with index range checks	572.0	658.1	664.2	561.1	902.5	507.6	1,550.1	1,689.9	1,958.1	1,726.6
k: Calling empty PROC procedure	424.2	412.4	467.8	558.8	733.2	743.3	1,329.9	1,356.3	1,580.1	2,754.7
l: like k but 4 parameter PROCEDURE	559.2	548.9	630.5	687.2	994.3	980.9	1,764.4	1,837.1	2,139.5	4,400.5
m: CARDINAL vector assignment (block move)	862.4	894.4	895.8	1,113.8	844.4	1,049.7	1,508.7	1,545.8	1,809.5	2,390.5
n: Linear list traversal	789.2	902.0	906.8	252.9	1,413.9	1,902.0	2,661.1	2,689.3	3,083.7	3,816.0
o: Read from file	106.1	98.7	118.0	110.1	109.5	83.3	196.5	198.0	229.6	295.9
Average performance	**512.4**	**572.1**	**585.4**	**446.0**	**983.6**	**1,008.8**	**1,803.9**	**1,896.1**	**2,146.9**	**2,673.9**
p: like e but LONGREAL (basic arithmetic)	1.825			1.592	71.342	70.058	132.675	133.742	154.558	257.597
q: like f but LONGREAL (transcendental fcts.)	2.023			1.586	17.293	20.532	31.232	31.829	37.084	61.806
p/e	0.924			0.960	1.000	0.996	1.082	0.999	1.000	0.855
q/f	0.758			0.791	1.002	0.999	1.084	1.006	1.001	0.932

References:

N. Wirth, 1981. The personal computer LILITH, IFI Bericht 40, Swiss Federal Institute of Technology Zurich, Zurich Switzerland, pp.66

Loeffler, T.J., Lischke, H., Fischlin, A. & Ulrich, M., 1995. Benchmark experiments on workstations. Systems Ecology Report No. 24, Institute of Terrestrial Ecology, Swiss Federal Institute of Technology ETH, Zurich, Switzerland, 40 pp.

Tabelle 9: SymDiff-Vergleich der Testsuite-Resulate P1 mit den Referenz-Outputs des P20 Kompilers. Für den Vergleich mit SymDiff wurde das Dataframe ETSSymDifFs_317vs314.DTF verwendet. Die Resultate geben die gefundenen Unterschiede in der Form Rt/Rz/Rl/Rr an, wobei Rt die Anzahl sich unterscheidender reeller Zahlenangaben ist, Rz die Anzahl reeller Zahlenangaben deren Differenz ungleich null ist, Rl die Anzahl reeller Zahlenangaben die sich in double precision um mehr als epsLR (1.110223E-016) unterscheiden und Rr die Anzahl reeller Zahlen die sich in single precision um mehr als epsSR (1.803317E-130) unterscheiden

Testoutput	**Resultat**	**Diagnose**
TstDMMLibs-0	-	NO FILE: Test existiert nicht
TstDMMLibs-1	-	NO FILE: Test existiert nicht
TstDMMLibs-2	-	NO FILE: Test existiert nicht
TstDMFloatEnv	-	ERROR: exception at startup
TstDMFiles	0/0/0/0	-
TstDMEntryF	0/0/0/0	-
TstDMEditFlds	0/0/0/0	-
TstDMConv-0	2/0/0/0	ERROR: real number invalid operation
TstDMConv-1	-	NO FILE: Programm abgestürzt
TstDMConv-2	-	NO FILE: Programm abgestürzt
TstDMConv-3	-	NO FILE: Programm abgestürzt
TstDMConv-4	-	NO FILE: Programm abgestürzt
Combined	0/0/0/0	-
Linked	0/0/0/0	-
LinkedS	-	FATAL - not yet implemented
CombineS	-	FATAL - not yet implemented
Diversity	848/848/848/848	von 3'506, Fehler nicht divergierend!
DiverseS	-	FATAL - not yet implemented
ForestYield	0/0/0/0	-
GauseIdentif	-	ERROR: real number invalid operation
ParamIter	-	NO FILE: Test existiert nicht
Insect	10/0/0/0	von 18'664
Logistic	0/0/0/0	-
LogGrowS	-	FATAL - not yet implemented
Lorenz	1/0/0/0	-
LotVolt3Ms	0/0/0/0	-
LVPhasePlot	0/0/0/0	-
Markov	4/4/4/4	von 53'257
Markov.Summary	0/0/0/0	-
Sensitivity	24/0/0/0	von 3'132
StochLogGrow	-	ERROR: real number invalid operation
SwissPop	1840/1840/1840/1840	von 8'860, nicht divergierend
UseTabFunc	89/89/89/89	von 206
VDPol	0/0/0/0	-
CarPollution	101/101/101/101	von 372
CoupledMstr	0/0/0/0	-
CoupldMS	-	FATAL - not yet implemented
GHMaster	549/549/549/549	von 819
LBM	160/160/160/160	von 482
LBMS	-	FATAL - not yet implemented

Tabelle 10: Die Tabelle zeigt den Vergleich der mit der P1-Implementation generierten Test-Resultate mit jenen der Compile20 Implementation.

```
================================================================================
Overall Statistics: 40 file pairs compared  ( 23 DONE )

Comparisons:
    Item                 Ntot compared     Ntot with differing tokens
    Lines                     31549                     2055
    Tokens                   305981                     5705
    Real numbers             146417                     3963

Similarity index psi = 1-sum(|r1-r2|)/sum(r1+r2):
    E[psi]     =  1.0000
    VAR[psi]   =  0.0000
    Coeff.var. =  0.0000
    MIN(psi)   =  1.0000
    MAX(psi)   =  1.0000
    n          =      20
    n illegal  =       0

Histograms of computed real number differences (excluding r1=NAN OR r2=NAN):
    d = (r1-r2)                            drel = 100*(r1-r2)/|r1|
    from        to          count          from        to          count
   -1.80E+308  -1.00E+000      23         -1.80E+308  -1.00E-001      15
   -1.00E+000  -1.00E-002      24         -1.00E-001  -1.00E-002      15
   -1.00E-002  -1.00E-004     535         -1.00E-002  -1.00E-003      65
   -1.00E-004  -1.00E-006     483         -1.00E-003  -1.00E-004     127
   -1.00E-006  -1.00E-008     583         -1.00E-004  -1.00E-005     736
   -1.00E-008  -1.00E-010       0         -1.00E-005  -1.00E-006     690
   -1.00E-010  -1.00E-012       0         -1.00E-006  -1.00E-007       0
   -1.00E-012  -1.00E-014       0         -1.00E-007  -1.00E-008       0
   -1.00E-014  -1.00E-016       0         -1.00E-008  -1.00E-009       0
   -1.00E-016   0.00E+000       0         -1.00E-009   0.00E+000       0
          0.00E+000        141140               0.00E+000        127681
    0.00E+000   1.00E-016       0          0.00E+000   1.00E-009       0
    1.00E-016   1.00E-014       0          1.00E-009   1.00E-008       0
    1.00E-014   1.00E-012       0          1.00E-008   1.00E-007       0
    1.00E-012   1.00E-010       0          1.00E-007   1.00E-006       0
    1.00E-010   1.00E-008       0          1.00E-006   1.00E-005     911
    1.00E-008   1.00E-006     380          1.00E-005   1.00E-004    1183
    1.00E-006   1.00E-004     679          1.00E-004   1.00E-003     121
    1.00E-004   1.00E-002    1192          1.00E-003   1.00E-002      45
    1.00E-002   1.00E+000      17          1.00E-002   1.00E-001       4
    1.00E+000   1.80E+308      10          1.00E-001   1.80E+308      14
    Total comparisons      145066          Total r1<>0            131607

Deviation statistics: Ntot dL/T/Rt/Rz/Rl/Rr = 2055/5705/3963/3926/3926/3667

L  - Lines
T  - Tokens
R  - Real numbers (REAL = single prec. or LONGREAL = double prec.)
Rt - Real tokens
Rz - ABS(real difference) > 0.0
Rl - ABS(real difference) > epsilonLR = 2.220446E-016 (LONGREAL, double prec.)
Rr - ABS(real difference) > epsilon   = 1.192093E-007 (REAL, single prec.)
All differences and comparisons were computed in expressions of double precision

================================================================================
```

6 Diskussion

Das Vorgehen bei der Portierung hängt von den Prioritäten ab. Es gibt keine Lösung, welche alle möglichen Aspekte am besten deckt. Aus den Resultaten dieser Arbeit drängt sich jedoch folgendes Vorgehen auf:

6.1 Portierung auf die Hauptplattform Mac OS X

Mit der Carbon Bibliothek, welche vom P1 Kompiler mitgeliefert wird, ist die Interaktion mit dem Mac OS X Betriebssystem voll gewährleistet. In Zukunft soll auch die Cocoa Bibliothek mit dem P1 zur Verfügung gestellt werden. Da P1 erst seit der Version 8 eine native Modula-2 Entwicklungsumgebung für Mac OS X zur Verfügung stellt, lässt die Integration des Kompilers in die OS X Umgebung in gewissen Bereichen, insbesondere bei den Prozessen der Kompilation, des Debuggings und des Linkens noch etwas zu wünschen übrig. Mit einer Integration des P1 Kompilers in die OS X Entwicklungsumgebung XTools ist in der nächsten Zeit zu rechnen. Herr Wiedemann, der Entwickler von P1 hat sich im Laufe dieser Arbeit als zuverlässiger Partner erwiesen und man kann davon ausgehen, dass weitere Anstrengungen von seiner Seite unternommen werden und dass er weiterhin auf Wünsche und Anregungen bezüglich der Weiterentwicklung des Kompilers eingehen wird.

Die Portierung von RAMSES Version BatchDM, auf den P1 Kompiler unter Mac OS X, ist bereits weit fortgeschritten. Erste Modelle wurden berechnet und deren Resultate mit Referenzdatensätzen verglichen. Zwar traten dabei noch verschiedene Probleme auf, doch zeichnete sich ab, dass mit korrekten Resultaten gerechnet werden kann.

Auch bezüglich Performance schneiden P1 Programme gut ab. Wie die Benchmark-Tests zeigen, sind P1 Programme in der Lage, die mit der G5-Architektur verüfgbare Leistungssteigerung v.a. auch in der Fliesskomma-Arithmetik auszunutzen. Eine Leistungsteigerung von einer Grössenordnung 4 - 5 im Vergleich zum MacMETH Kompiler in der Classic-Umgebung ist möglich. Verglichen mit der EPC-Implementation auf der Sun ist die P1-DM bei vergleichbarer Taktrate um Faktor 2 - 3 schneller. Der zur Zeit leistungsfähigste PPC Rechner (G5, 1.6 GHz) ist in der Gesamtperformanz um Faktor 6 schneller als die schnellste verfügbare Sun (Blade 100, 500 MHz)

Die Anpassungen des Codes an die ISO Sprachdefinition wurde mit der Portierung der DM grösstenteils abgeschlossen. Für die Portierung der interaktiven DialogMaschine müssten weiter in einer beschränkten Anzahl von DM-Modulen Interaktionen mit dem Betriebssystem von Toolbox-Aufrufen zu Aufrufen von Carbon-Schnittstellen umgeschrieben werden. Da sich diese Low-Level Prozeduren fast ausschliesslich in ein paar wenigen Modulen der DM konzentrieren, könnte diese Portierung relativ einfach von statten gehen.

Alles in allem erweist sich P1 unter Mac OS X als eine durchaus interessante Alternative zu den bisherigen RAMSES Hauptplattformen Classic (interaktive DM) und Sun (BatchDM). Mit der gleichen Hauptplattform für beide RAMSES Versionen könnte der Aufwand für die Wartung des Codes minimiert werden. Durch die ISO-Kompatibilität des auf den P1 Kompiler portierten Codes, erhöht sich die Portabilität der Software generell. Nebst der weiterhin uneingeschränkten Unterstützung verschiedener PIM-Varianten, lässt sich der P1-kompatible Code mit reduziertem Aufwand auf weitere ISO-Kompiler (z.B. Stony Brook, XDS-x86 und XDS-C) portieren.

6.2 Generische Lösung für weitere Zielplattformen

Hierbei geht es in erster Linie darum eine Lösung für die Portierung der RAMSES Software auf die Plattformen Solaris/SPARC und Windows zu finden. Eine Linux Unterstützung wird nicht explizit gefordert, wäre aber wegen der zunehmenden Popularität von Linux durchaus wünschenswert. Da der

Unterhalt der verschiedenen Ports viel Zeit in Anspruch nimmt, müsste nach einer generischen Lösung gesucht werden, welche die Anzahl der zu pflegenden Ports reduziert, im Idealfall auf eine RAMSES Version.

Neu-Implementation in Java

Eine Möglichkeit der plattformunabhängigen Implementation von RAMSES wäre die Implementation in Java, einer plattformunabhängigen und zur Zeit sehr populären Programmiersprache. Der vom Java-Kompiler produzierte Byte-Code kann auf jeder Zielplattformen entweder direkt interpretiert, oder für die Steigerung der Performanz in Maschinensprache übersetzt werden. Dabei muss darauf geachtet werden, dass auch die Implementierung plattformunäbhängig erfolgt, d.h. keine plattformspezifischen Bibliotheken verwendet werden. Der Aufwand für eine solche Neuimplementation wäre jedoch sehr gross. Die ganze RAMSES Software müsste neu geschrieben werden, wobei auf Grund objektorientierter Spracheigenschaften von Java auch Änderungen in der Architektur von RAMSES erfolgen müssten. Das Resultat wäre insofern vielversprechend, dass es sich bei Java um eine aktuelle Programmiersprache mit Zukunftsaussichten handelt und nur noch eine RAMSES-Version gewartet werden müsste. Bezüglich den Qualitäten von Java für das wissenschaftliche Rechnen müssten weitere Analysen v.a. bezüglich Performance und Genauigkeit gemacht werden. Insbesondere müssten Simulationen auf allen Plattformen dasselbe Resultat ergeben.

Verwendung von Standard Bibliotheken

Der grösste Teil des RAMSES Codes ist bereits plattformunabhängig, d.h. er kann für alle bisherigen Implentationen ohne Modifikation verwendet werden. Das trifft v.a. auf die Module höherer Schichten zu. Einige Module sind nur partiell plattformabhängig, d.h. gewisse Teile des Codes werden mit Hilfe von Kompilerflags konditionell kompiliert, meist handelt es sich dabei um Unterschiede in der Sprachdefinition, seltener um Unterschiede bezüglich Bibliotheken. Nur wenige Module sind voll plattformabhängig. Diese liegen in mehreren Versionen vor und müssen demnach auch in mehreren Versionen gewartet werden, in diesem Fall liegen meist grosse Unterschiede in den verwendeten APIs vor. Die Plattformabhängigkeit betrifft v.a. Module tieferer Schichten, insbesondere SysDep und teilweise die DialogMaschine.

Im Allgemeinen kann schon jetzt der Aufwand für die Wartung verschiedener Ports als gering bezeichnet werden. Minimieren könnte man den Wartungsaufwand der jetzigen RAMSES Implementation, bei gleichbleibender Portabilität v.a. durch die Verwendung standardisierter Bibliotheken in Modulen der DialogMaschine. Das betrifft in erster Linie Bibliotheken für die grafische Benutzeroberfläche der RAMSES-Software. Standardisierte Bibliotheken liegen in der Regel als C-Schnittstellen vor und erfordern daher die Möglichkeit des Kompilers, C-Schnittstellen anzusprechen. Folgende, für eine Vielzahl von Plattformen verfügbare Grafikbibliotheken könnten dafür in Frage kommen:

- wxWidgets für Fensterverwaltung (www.wxwindows.org)
- X (www.xfree86.org)
- GTK+ (www.gtk.org)
- Qt (www.trolltech.com)
- OpenGL für 2D und 3D Grafik

6.3 Aussicht

Kurzfristig zeichnet sich mit der Portierung der RAMSES Software Version BatchDM auf den P1 Kompiler unter Mac OS X eine leistungsfähige RAMSES-Implementation auf einem modernen Arbeitsplatzrechner ab. Die Erfahrungen, welche im Rahmen dieser Arbeit gesammelt wurden zeigen, dass mit dem P1 Kompiler eine gute Wahl getroffen wurde und sich weitere Portierungsanstrengungen in diese Richtung auszahlen werden. Mit grösseren Hürden ist dabei nicht mehr zu rechnen.

Mittelfristig zeichnen sich auch für die anderen Plattformen Lösungen ab. Im Auge behalten sollte man den GM2 Kompiler, welcher bereits in naher Zukunft für eine grössere Anzahl Plattformen zur Verfügung stehen könnte. Vorbereitend für weitere Portierungen könnte die Verwendung standardisierter Grafikbibliotheken Sinn machen. Ein besonderes Augenmerk sollte dort auf die Bibliotheken des X-Servers geworfen werden, welcher mittlerweile für alle gängigen Plattformen zur Verfügung steht und insbesondere in der UNIX/Linux-Welt eine zentrale Rolle einnimmt.

Langfristig sollte man sich über die Zukunft der Programmiersprache Modula-2 Gedanken machen. Die Auswahl an Kompilern ist bereits jetzt erschreckend gering, häufig steht pro Plattform nur ein Kompiler zur Auswahl (z.B. P1 für Mac OS X). Auch die Entwicklergemeinden, welche hinter den Kompilern stecken sind meist erschreckend klein (je nur eine Person bei den vielversprechendsten Kompilern P1 und GM2). Auch das Angebot an Konvertern z.B. von Modula-2 nach C oder nach Java deutet an, dass Modula-2 in der nächsten Zeit noch mehr an Bedeutung verlieren wird und kaum Zukunftschancen hat. Um eine Neuimplementation z.B. in Java (objektorientiert) oder in C (prozedural) kommt man kaum herum, will man die RAMSES-Software langfristig am Leben erhalten.

Literatur

[1] Black, Norman, 2004. *Stony Brook Modula-2 Compiler Build 28.* http://www.home.ix.netcom.com/~stonybrk/ (last accessed: 13. April 2004)

[2] Borchert H. und Hasch M., 2000.*Ulm's Modula-2 Compiler 3.0b8.* http://www.mathematik.uni-ulm.de/modula/ (last accessed: 13. April 2004)

[3] Edinburgh Portable Compiler Ltd., 1991. *EPC Modula-2 2.0.8.* Homepage closed, manpage: http://www.ito.umnw.ethz.ch/SysEcol/SimSoftware/RAMSES/em2_2.0.6.html (last accessed: 13. April 2004)

[4] Excelsior, LLC, 2001. *Native XDS-x86 2.51.* http://www.excelsior-usa.com/xdsx86.html (last accessed: 13. April 2004)

[5] Excelsior, LLC, 2001. *XDS-C.* http://www.excelsior-usa.com/xdsc.html (last accessed: 13. April 2004)

[6] Fischlin, A., 2003. *Ramses und OS X Technische Semesterarbeit Oliver Gardi.* Unveröffentlicht

[7] Fischlin, A. & Gyalistras, D., 2004. *Anhang - Kriterien zur Beurteilung einer Modula-2 Implementation.* Unveröffentlicht

[8] Fischlin, A., 1991. *Interactive Modeling and Simulation of Environmental Systems on Working Stations.* In: Möller, D.P.F. & Richter, O. (eds.), Analysis of dynamic systems in medicine, biology, and ecology. Informatik-Fachberichte 275: 131-145.

[9] Fischlin, A., Gyalistras, D. & Löffler, T.J., 1995. *Installation Guide and Technical Reference of the RAMSES Software (Version 2.2) For Apple Macintosh Computers.* Technical Systems Ecology Report, Swiss Federal Institute of Technology, Zürich, 44pp.

[10] Fischlin, A. et al., 1994. *ModelWorks 2.2 An Interactive Simulation Environment for Personal Computers and Workstations.* Technical Systems Ecology Report Nr. 14, Swiss Federal Institute of Technology, Zürich, 324pp.

[11] Gardi, O. & Fischlin, A., 2004. *Zeitplan und Vorgehensweise - Evaluation von Portierungsoptionen der Dialog-Maschine auf die Systeme MacOS X und UNIX/Linux.* Unveröffentlicht

[12] GM2-Community, 2004. *GNU Modula-2 0.43.* http://floppsie.comp.glam.ac.uk/Glamorgan/gaius/web/GNUModula2.html (last accessed: 13. April 2004)

[13] GMD research laboratory Karlsruhe, 1999. *MOCKA (MOdula Compiler KArlsruhe).* http://www.info.uni-karlsruhe.de/~modula/ (last accessed: 26. April 2004)

[14] Löffler, T. J., Fischlin, A., Ulrich, M., 1996. *Benchmark Experiments on Workstations.* Technical Systems Ecology Report Nr. 24, Swiss Federal Institute of Technology, Zürich, 53pp.

[15] MHCCorp.com, 2001. *Canterbury Modula-2 2.5.35.* http://www.mhccorp.com/modula-2.html (last accessed: 13. April 2004)

[16] P1 Gesellschaft für Informatik GmbH, 2004. *P1 Modula 7.3.* http://www.awiedemann.de/compiler/index.html (last accessed: 13. April 2004)

[17] PLAS, Queensland University of Technology, 1996 2004. *Gardens Point Modula (GPM).* http://www.citi.qut.edu.au/research/plas/projects/gardens_point_modula.jsp (last accessed: 26. April 2004)

[18] Wirth, N. et al., 1992. *MacMETH 3.2 User Manual - A fast Modula-2 Language System for the Apple Macintosh.* Departement Informatik, Swiss Federal Institute of Technology, Zürich, 116pp.

Anhang

A Skripts zur Analyse des Quelltext

A.1 Auswahl der Module – jobs.sh

Die folgenden Befehle wurden ausgeführt, um die Zustandsanalyse mit Hilfe der UNIX Shell Skripts `analyse_code.sh` (Anhang A.2) und `analyse_quick-refs.sh` (Anhang A.3) durchzuführen. Die unten aufgeführten Befehle sind relativ banal und hier nur der Vollständigkeit halber aufgeführt. Von Bedeutung sind die Pfade der untersuchten Packages und Subpackages, sowie die Include- bzw. Exclude-Kriterien, die für die Analyse angewendet wurden.

Sollte das Skript in einer anderen Umgebung ausgeführt werden, müssen die Pfade `$Dev` für den SED Stammordner, `$Bin` für den Skript-Ordner, `$results` für die Log-Dateien und `$RMSDoc` für die QuickReferences, angepasst werden.

```
#! /bin/sh

Dev="/Users/og/work/semesterarbeit/SED/Dev"
Bin="/Users/og/work/semesterarbeit/sa_bin"
results="$Bin/results"

AuxLibSE="$Dev/AuxLibDevSE"
AuxLibPD="$Dev/AuxLibDevPD"
AuxLib="$AuxLibSE $AuxLibPD"
M2="$Dev/MacMETHDev"
DMLibDev="$Dev/DMLibDev"
DMHigh="$DMLibDev/High.*"
PCDevDM="$Dev/PCDev/DM"
DM="$DMLibDev $PCDevDM"
RASS="$Dev/RASSDev"
BatchDM="$Dev/RASSDev/BatchDMDev"
SysDep="$Dev/RASSDev/SysDep"
MW="$Dev/RAMSESDev/MWLibDev"
EMW="$Dev/RAMSESDev/EMWDev"
PALibDev="$Dev/RAMSESDev/PALibDev"
RMSLibDev="$Dev/RAMSESDev/RMSLibDev"
RMSShellDev="$Dev/RAMSESDev/RMSShellDev"
RMS="$PALibDev $RMSLibDev $RMSShellDev"
SciLib="$Dev/SciLibDev"
ISIS="$Dev/ISISLibDev"
Tools="$Dev/ToolsDev"

echo checking AuxLibSE ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $AuxLibSE \
        2>&1 > $results/check_AuxLibSE.txt
echo checking AuxLibPD ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $AuxLibPD \
        2>&1 > $results/check_AuxLibPD.txt
echo checking AuxLib ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $AuxLib \
        2>&1 > $results/check_AuxLib.txt
echo checking M2 ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $M2 \
        2>&1 > $results/check_M2.txt
echo checking DMLibDev ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $DMLibDev \
        2>&1 > $results/check_DMLibDev.txt
echo checking DMHigh ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $DMHigh \
        2>&1 > $results/check_DMHigh.txt
echo checking PCDevDM ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $PCDevDM \
        2>&1 > $results/check_PCDevDM.txt
```

```
echo checking DM ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $DM \
        2>&1 > $results/check_DM.txt
echo checking RASS ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $RASS \
        2>&1 > $results/check_RASS.txt
echo checking BatchDM ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $BatchDM \
        2>&1 > $results/check_BatchDM.txt
echo checking SysDep ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $SysDep \
        2>&1 > $results/check_SysDep.txt
echo checking MW ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $MW \
        2>&1 > $results/check_MW.txt
echo checking EMW ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $EMW \
        2>&1 > $results/check_EMW.txt
echo checking PALibDev ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $PALibDev \
        2>&1 > $results/check_PALibDev.txt
echo checking RMSLibDev ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $RMSLibDev \
        2>&1 > $results/check_RMSLibDev.txt
echo checking RMSShellDev ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $RMSShellDev \
        2>&1 > $results/check_RMSShellDev.txt
echo checking RMS ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $RMS \
        2>&1 > $results/check_RMS.txt
echo checking SciLib ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $SciLib \
        2>&1 > $results/check_SciLib.txt
echo checking AuxLib ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $ISIS \
        2>&1 > $results/check_ISIS.txt
echo checking ISIS ...
$Bin/analyse_code.sh -v -e "/.* old.*|/.*Test.*/| DevAux/" $Tools \
        2>&1 > $results/check_Tools.txt
echo checking Dev ...
$Bin/analyse_code.sh -v -e "/.* old.*| DevAux/|/XrefTest\.DEF" $Dev \
        2>&1 > $results/check_Dev.txt
echo checking Testing ...
$Bin/analyse_code.new.sh -v -i "/.*[T|t]est.*/" -e "/.* (old|DevAux).*/|/XrefTest\.DEF" $Dev \
        2>&1 > $results/check_Testing2.txt

RMSDoc="/Users/og/work/semesterarbeit/RMS/Docu"
AuxLibQR="$RMSDoc/AuxLib/AuxLib Quick Reference nc.txt"
DMQR="$RMSDoc/DM/DM Quick Reference nc.txt"
ISISQR="$RMSDoc/ISIS/ISIS Quick Reference nc.txt"
MWQR="$RMSDoc/MW/MW Quick Reference nc.txt"
SciLibQR="$RMSDoc/ScienceLib/ScienceLib Quick Reference nc.txt"

echo checking AuxLib QuickRefs ...
$Bin/analyse_quick-refs.sh "$AuxLibQR" \
        2>&1 > $results/QR-AuxLib.txt
echo checking DM QuickRefs ...
$Bin/analyse_quick-refs.sh "$DMQR" \
        2>&1 > $results/QR-DM.txt
echo checking ISIS QuickRefs ...
$Bin/analyse_quick-refs.sh "$ISISQR" \
        2>&1 > $results/QR-ISIS.txt
echo checking MW QuickRefs ...
$Bin/analyse_quick-refs.sh "$MWQR" \
        2>&1 > $results/QR-MW.txt
echo checking SciLib QuickRefs ...
$Bin/analyse_quick-refs.sh "$SciLibQR" \
        2>&1 > $results/QR-SciLib.txt
```

A.2 Analyse des Quelltextes – analyse_code.sh

Dieses UNIX Shell Skript wurde dazu verwendet, die Kennzahlen der RAMSES Software aus dem Source-Code zu ermitteln. Die Dokumentation des Skripts ist dem unten aufgeführten Source-Code zu entnehmen.

```
#! /bin/bash

# Skript fuer die Ermittlung des Umfangs eines Modula 2 Codebestandes
# ------------------------------------------------------------------
#
# Unterhalb eines oder mehrerer Verzeichnisse werden all jene Modula 2
# Dateien (.DEF, .MOD, .OBM) welche nicht durch eine vom  Benutzer de-
# finierte "Regular Expression" ausgeschlossen werden bezueglich fol-
# genden Kriterien analysiert:
#
# .DEF & .MOD:  - Anzahl Module .DEF und .MOD
#               - Anzahl Zeilen mit Kommentar und ohne Leerzeilen
#               - Anzahl Zeilen ohne Kommentar und ohne Leerzeilen
#
# .DEF:         - Anzahl exportierter Prozeduren
#               -   ""        ""      Typen
#               -   ""        ""      Konstanten
#               -   ""        ""      Variablen
#
# .OBM:         - Anzahl Binrdateien.OB
#               - Anzahl kByte
#
# Optionen und Parameter:
#
# -v verbose; zeigt an welche Dateien von der Analyse ausge-
# ausgeschlossen werde, sowie die gewonnen Informationen
# aller analysierten Dateien (siehe oben).
#
# -i "PATTERN" include; filtert die Dateipfade und schliesst alle Da-
# teien aus, die nicht dem definierten Regulaeren Aus-
# druck entsprechen (siehe 'man egrep'). Diese Option
# entspricht also dem Wortlaut "nur".
#
#
# -e "PATTERN" exclude; filtert die Dateipfade entsprechend dem
#               definierten Regulaeren Ausdruck (siehe 'man egrep'). Da-
# teien deren Pfad dem PATTERN entsprechen werden von der
# Analyse ausgeschlossen. Diese Option entspricht dem Wort-
# laut "ausser".
#
#
# Technische Semesterarbeit
# Oliver Gardi, Gruppe Systemoekologie
# Institut fuer terrestrische Oekologie
# Eidgenssische Technische Hochschule
#
# Letzte Aenderung: 10.4.2004

# obsolet: Konstanten (Pfade je nach Umgebung anpassen)
# BIN_DIR=~/work/semesterarbeit/sa_bin
# DEV_DIR=~/work/semesterarbeit/SED/Dev


# Variablen initialisieren, bzw. Defaultwerte setzen
# --------------------------------------------------

log_verbose=0
include_RE="*"
exclude_RE="^.*/$"

code=""
modules=""
```

```
tmp_file_cnt=0
tmp_ln_cnt=0
tmp_ln_cnt_nc=0

def_cnt=0
def_ln_cnt=0
def_ln_cnt_nc=0
proc_count=0
type_cnt=0
const_cnt=0
var_cnt=0

mod_cnt=0
mod_ln_cnt=0
mod_ln_cnt_nc=0

obm_cnt=0
obm_kb=0

echo "##############################################################################"
echo "# $0 $@ "
echo "##############################################################################"
echo

# Parameter abfragen
# ------------------

while getopts ":vi:e:" opt
do
  case $opt in
    v ) log_verbose=1 ;;
    i ) include_RE="$OPTARG" ;;
    e ) exclude_RE="$exclude_RE|$OPTARG" ;;
    \?) echo "usage: analyse_code.sh [-v] [-i incl_RE] [-e excl_RE] folders"
        echo -n "example: analyse_code.sh -v -i \"PATTERN\" "
echo -n "-e \"/.*( old| Test).*/|/.* DevAux.*/\" "
echo "../SED/Dev/AuxLibDevPD ../SED/Dev/AuxLibDevPD"
        exit 1
  esac
done
shift $(($OPTIND - 1))


# ===============
# Hilfsprozeduren
# ===============

# Log-Prozedur fuer Verbose Modus
# -------------------------------

verbose () {
  if [ $log_verbose -eq 1 ]
  then
    if [ "$1" == "-n" ]
    then
      echo -n "$2"
    else
      echo "$1"
    fi
  fi
}


# Prozedur entfernt alle leeren Zeilen aus der Variablen $Code
# ------------------------------------------------------------

remove_blank_lines () {
  code=‘echo $code | \
           perl -pe 's/#newline#(\s*#newline#)*\s*#newline#/#newline#/g’‘
}
```

```
# Prozedur entfernt alle Kommentare aus der Variablen $Code
# ---------------------------------------------------------

remove_comment () {
  code=‘echo $code | \
           perl -pe ’s/\{|\}//g’       | \
         perl -pe ’s/"(\(\*|\*\))"//g’ | \
    perl -pe ’s/\(\*/\{/g’      | \
           perl -pe ’s/\*\)/\}/g’‘

# old and obsolete filters:
#            perl -pe ’s/\s*\(\*/\{/g’      | \
#            perl -pe ’s/\*\)\s*/\}/g’‘
#          perl -pe ’s/"[^"]*"/String/g’ |\

  # Entferne auch verschachtelte Kommentare
  while echo $code | grep ’{’ > /dev/null
  do
    code=‘echo $code | perl -pe ’s/\{[^{}]*\}//g’‘
  done

  remove_blank_lines
}


# Prozedur bereitet die Datei fr die folgede Analyse vor und zaehlt Zeilen
# ------------------------------------------------------------------------

count_lines () {

    verbose -n "processing: $1 ->"

    # Datei in UNIX Format umwandeln und in Variable lesen (\n -> "\newline")
    # Der Mac Linefeed schreibt sich ’Control-v Control-m’ oder \r
    code=‘cat "$1" | tr "\r" "\n" | perl -pe ’s/\n/#newline#/g’‘

    # leere Zeilen entfernen
    remove_blank_lines

    tmp_ln_cnt=‘echo $code | perl -pe ’s/#newline#/\n/g’ | wc -l‘
    verbose -n " l(+): $(($tmp_ln_cnt))"

    # Kommentar entfernen
    remove_comment

    # Code Zeilen zhlen
    tmp_ln_cnt_nc=‘echo $code | perl -pe ’s/#newline#/\n/g’ | wc -l‘
    verbose -n ", l(-): $(($tmp_ln_cnt_nc))"
}


# Prozedur filtert Module nach Regexp in Parameter und zaehlt Module
# ------------------------------------------------------------------

count_modules () {
  modules=‘echo $modules | perl -pe ’s/\s/#newline#/g’‘
  verbose "excluding files: ‘echo "$modules" | tr "%" " " | \
           perl -pe ’s/#newline#/\n/g’ | egrep -v "$include_RE" | \
     egrep "$exclude_RE"‘"
  modules=‘echo "$modules" | tr "%" " " | perl -pe ’s/#newline#/\n/g’ | \
           egrep "$include_RE" | egrep -v "$exclude_RE" | tr " " "%"‘
  tmp_file_cnt=‘echo $modules | wc -w‘
}


# ==========================================================
# Prozedur zum Finden und Analysieren der Definition Modules
# ==========================================================

check_defs () {
```

```
  verbose ""
  verbose "******* Checking DEFs *******"
  modules=`find $1 -type f -name  "*.[D|d][E|e][F|f]" -print | tr " " "%"`
  count_modules
  def_cnt=$(($def_cnt + $tmp_file_cnt))
  verbose ""
  for module in $modules
  do
    count_lines "`echo $module | tr "%" " "`"
    def_ln_cnt=$(($def_ln_cnt + $tmp_ln_cnt))
    def_ln_cnt_nc=$(($def_ln_cnt_nc + $tmp_ln_cnt_nc))

    # Bezeichner zaehlen
    # ------------------

    # #newline# entfernen
    code=`echo $code | perl -pe 's/(\s*#newline#)+\s*/ /g'`

    # Klammern mit Parametern entfernen
    while echo $code | grep '(' > /dev/null
    do
      code=`echo $code | perl -pe 's/\([^()]*\)//g'`
    done

    # RECORD ... END entfernen
    code=`echo $code | \
           perl -pe 's/RECORD/\{/g' | \
           perl -pe 's/END/\}/g'`
    while echo $code | grep '{' > /dev/null
    do
      code=`echo $code | perl -pe 's/\{[^{}]*\}//g'`
    done

    # obsolet: PROCEDURE() maskieren
    # code=`echo $code | \
    #       perl -pe 's/PROCEDURE\(\)/PROC\(\)/g'`

    # Code ordnen
    code=`echo $code | \
          perl -pe 's/PROCEDURE |VAR |CONST |TYPE/#newline#$&/g'`

    # Bezeichner zaehlen(kumulativ)
    local tmp_count=`echo $code | perl -pe 's/#newline#/\n/g' \
                     | grep 'PROCEDURE ' | wc -l`
    verbose -n ", p: $(($tmp_count))"
    proc_cnt=$(($proc_cnt + $tmp_count))

    local tmp_count=`echo $code | perl -pe 's/#newline#/\n/g' | grep 'VAR ' | \
                     tr ";" "\n" | perl -pe 's/^\s*$//g' | wc -l`
    verbose -n ", v: $(($tmp_count))"
    var_cnt=$(($var_cnt + $tmp_count))

    local tmp_count=`echo $code | perl -pe 's/#newline#/\n/g' | grep 'CONST ' | \
                     tr ";" "\n" | perl -pe 's/^\s*$//g' | wc -l`
    verbose -n ", c: $(($tmp_count))"
    const_cnt=$(($const_cnt + $tmp_count))

    local tmp_count=`echo $code | perl -pe 's/#newline#/\n/g' | grep 'TYPE ' | \
                     tr ";" "\n" | perl -pe 's/^\s*$//g' | wc -l`
    verbose ", t: $(($tmp_count))"
    type_cnt=$(($type_cnt + $tmp_count))

  done
}


# ============================================================
# Prozedur zum Finden und Analysieren der Implementation Module
# ============================================================

check_mods () {
```

```
  verbose ""
  verbose "******* Checking MODs *******"
  modules=`find $1 -type f -name  "*.[M|m][O|o][D|d]" -print | tr " " "%"`
  count_modules
  mod_cnt=$(($mod_cnt + $tmp_file_cnt))
  verbose ""
  for module in $modules
  do
    count_lines "`echo $module | tr "%" " "`"
    mod_ln_cnt=$(($mod_ln_cnt + $tmp_ln_cnt))
    mod_ln_cnt_nc=$(($mod_ln_cnt_nc + $tmp_ln_cnt_nc))
    verbose ""
  done
}


# ================================================
# Prozedur zum Finden und Analysieren der Binaerdateien
# ================================================

check_obms () {
  verbose ""
  verbose "******* Checking OBMs *******"
  modules=`find $1 -type f -name  "*.[O|o][B|b][M|m]" -print | tr " " "%"`
  count_modules
  obm_cnt=$(($obm_cnt + $tmp_file_cnt))
  verbose ""
  for binary in $modules
  do
    binary=`echo $binary | tr "%" " "`
    local tmp_bytes=`du -k "$binary" | awk '{print $1}'`
    verbose "processing: $binary -> $(($tmp_bytes))"
    obm_kb=$(($obm_kb + $tmp_bytes))
  done
}


# ============================================
# Prozedur zum scannen verschiedener Verzeichnisse
# ============================================

scan () {
  echo ==========================================
  for folder in $*
  do
    echo checking folder: $folder ...
    # echo -n "checking defs ...   "
    check_defs $folder
    # echo ok
    # echo -n "checking mods ...   "
    check_mods $folder
    # echo ok
    # echo -n "checking obms ...   "
    check_obms $folder
    # echo ok
    echo ==========================================
  done
  echo
  echo Definition Modules .DEF
  echo -------------------------
  echo "# DEF modules:        $(($def_cnt))"
  echo "# lines total:        $(($def_ln_cnt))"
  echo "# lines no comment:   $(($def_ln_cnt_nc))"
  echo "# procedures:         $(($proc_cnt))"
  echo "# constants:          $(($const_cnt))"
  echo "# variables:          $(($var_cnt))"
  echo "# types:              $(($type_cnt))"
  echo
  echo Implementation Modules .MOD
  echo -------------------------
  echo "# MOD modules:        $(($mod_cnt))"
```

```
  echo "# lines total:        $(($mod_ln_cnt))"
  echo "# lines no comment:   $(($mod_ln_cnt_nc))"
  echo
  echo Binaries .OBM
  echo --------------------------
  echo "# OBM modules:        $(($obm_cnt))"
  echo "# kBytes:             $(($obm_kb))"
  echo
  echo ========================================
}

# Main
# ====

scan $*

# To-Do: Liste mit Packagenamen
#case $1 in
  # AuxLib | AuxLibDev )      scan AuxLibDevSE AuxLibDevPD ;;
  # AuxLibSE | AuxLibDevSE )  scan AuxLibDevSE ;;
  # AuxLibPD | AuxLibDevPD )  scan AuxLibDevPD ;;
  # M2* | MacMETH* )          scan MacMETHDev ;;
  #* )                       scan $1 ;;
#esac
```

A.3 Analyse der Schnittstellen – analyse_quick-refs.sh

Dieses Shell Skript zählt die in einer QuickReference Datei aufgeführten Definition Modules und analysiert die von dem Packages veröffentlichten Bezeichner. Vor der Analyse muss Nicht-M2-Kommentar (Kommentar ohne "(* *)") manuell entfernt werden.

Der Code lehnt sich sehr stark an der Analyse der .DEFs im Skript `analyse_code.sh` (Anhang A.2) an.

```
#! /bin/bash

code=""

mod_cnt=0
proc_cnt=0
type_cnt=0
const_cnt=0
var_cnt=0

# Prozedur entfernt alle leeren Zeilen aus der Variablen $Code
# -----------------------------------------------------------

remove_blank_lines () {
  code=‘echo $code | \
          perl -pe 's/#newline#(\s*#newline#)*\s*#newline#/#newline#/g’‘
}

# Prozedur entfernt alle Kommentare aus der Variablen $Code
# ---------------------------------------------------------

remove_comment () {
  code=‘echo $code | \
           perl -pe 's/\{|\}//g'       | \
           perl -pe 's/"(\(\*|\*\))"//g' | \
           perl -pe 's/\(\*/\{/g'      | \
           perl -pe 's/\*\)/\}/g’‘
  # Entferne auch verschachtelte Kommentare
  while echo $code | grep '{' > /dev/null
  do
    code=‘echo $code | perl -pe 's/\{[^{}]*\}//g’‘
  done
```

```
  remove_blank_lines
}

# Prozedur bereitet die Datei fuer die folgede Analyse vor und zaehlt Zeilen
# ----------------------------------------------------------------------------

prepare_code () {

    # Datei in UNIX Format umwandeln und in Variable lesen (\n -> "\newline")
    # Der Mac Linefeed schreibt sich 'Control-v Control-m' oder \r
    code=`cat "$1" | tr "\r" "\n" | perl -pe 's/\n/#newline#/g'`

    # leere Zeilen entfernen
    remove_blank_lines

    # Kommentar entfernen
    remove_comment
}

# Zaehlen und Analysieren der Definition Modules
# ==============================================================

check_qr () {
  mod_cnt=`cat "$1" | tr "\r" "\n" | \
                  egrep "\(\*[=]+[ ]+[A-Za-z0-9]+[ ]+[=]+\*\)" | wc -l`

  prepare_code "$1"

  # Bezeichner zaehlen
  # ------------------

  # #newline# entfernen
  code=`echo $code | perl -pe 's/(\s*#newline#)+\s*/ /g'`

  # Klammern mit Parametern entfernen
  while echo $code | grep '(' > /dev/null
  do
    code=`echo $code | perl -pe 's/\([^()]*\)//g'`
  done

  # RECORD ... END entfernen
  code=`echo $code | \
         perl -pe 's/RECORD/\{/g' | \
         perl -pe 's/END/\}/g'`
  while echo $code | grep '{' > /dev/null
  do
    code=`echo $code | perl -pe 's/\{[^{}]*\}//g'`
  done

  # Code ordnen
  code=`echo $code | \
        perl -pe 's/PROCEDURE |VAR |CONST |TYPE/#newline#$&/g'`

  # Bezeichner zaehlen(kumulativ)
  local tmp_count=`echo $code | perl -pe 's/#newline#/\n/g' \
                  | grep 'PROCEDURE ' | wc -l`

  proc_cnt=$(($proc_cnt + $tmp_count))

  local tmp_count=`echo $code | perl -pe 's/#newline#/\n/g' | grep 'VAR ' | \
                  tr ";" "\n" | perl -pe 's/^\s*$//g' | wc -l`
  var_cnt=$(($var_cnt + $tmp_count))

  local tmp_count=`echo $code | perl -pe 's/#newline#/\n/g' | grep 'CONST ' | \
                  tr ";" "\n" | perl -pe 's/^\s*$//g' | wc -l`
  const_cnt=$(($const_cnt + $tmp_count))

  local tmp_count=`echo $code | perl -pe 's/#newline#/\n/g' | grep 'TYPE ' | \
                  tr ";" "\n" | perl -pe 's/^\s*$//g' | wc -l`
  type_cnt=$(($type_cnt + $tmp_count))
```

```
}

scan () {
  check_qr "$1"
  echo ========================================
  echo Identifiers in Quickref $1
  echo ----------------------------------------
  echo "# modules:            $(($mod_cnt))"
  echo "# procedures:         $(($proc_cnt))"
  echo "# constants:          $(($const_cnt))"
  echo "# variables:          $(($var_cnt))"
  echo "# types:              $(($type_cnt))"
  echo ========================================
}

# Main
# ====

scan "$1"
```

B Benchmark-Programme

B.1 Integer Operationen – Sieve.mod

Dieses einfache Benchmark Programm für Integeroperationen berechnet 1'000 mal alle Primzahlen zwischen 0 und 8190. Dazu verwendet es den Algorithums des "Sieb des Erathostenes".

```
MODULE sieve;
(*(* $R-*) (* no subrange and arithmetic overflow test *)
(* $T-*) (* no index test (arrays, case) *)
(* $S-*) (* no stack overflow test *)*)



CONST Size=8190;

VAR Flags: ARRAY[0..Size] OF BOOLEAN;
  i,prime,k,count,iter: CARDINAL;
  ch: CHAR;

BEGIN (*sieve*)
  FOR iter:=1 TO 1000 DO
    count:=0;
    FOR i:=0 TO Size DO Flags[i]:=TRUE END;
    FOR i:=0 TO Size DO
      IF Flags[i] THEN
        prime:=i*2+3;
        k:=i+prime;
        WHILE k<=Size DO
          Flags[k]:=FALSE;
          INC(k,prime);
        END(*WHILE*);
        INC(count)
      END(*IF*)
    END(*FOR*);
  END(*FOR*);
END sieve.
```

B.2 Gleitkomma Operationen – Ereal.mod

Dieses einfache Benchmark Programm für Gleitkommaoperationen berechnet 50 mio. mal einen Term, bestehend aus einer Multiplikation, einer Addition und einer Division zweier Gleitkommazahlen.

```
MODULE ereal;

  (* e:  Basic floating point operations *)

  (*(* $T+ no Array index and case label boundary tests*)
  (* $P+ no procedure entry and exit code*)
  (* $S+ no Stack overflow test*)*)

  (* <* RangeCheck ( FALSE ) *>
  <* OverflowCheck ( FALSE ) *>
  <* IndexCheck ( FALSE ) *> *)

  VAR
    n:  INTEGER;

  PROCEDURE Test;
    VAR k:  INTEGER;
      r0, r1, r2: REAL;

  BEGIN (*Test*)
    k := 5000;
    r1 := 7.28;
```

```
    r2 := 34.8;
    REPEAT
      k := k-1;
      r0 := (r1*r2) / (r1+r2);
    UNTIL k = 0;
  END Test;

BEGIN (*ereal*)
    n:= 0;
    REPEAT
      n := n+1;
      Test;
    UNTIL n >= 10000;
END ereal.
```

B.3 DM-Benchmarks – Benchmark.mod

Das Skript Benchmarks.mod testet verschiedene DM-Implementationen auf deren Performanz. Getestet werden die von Wirth (1981) vorgeschlagenen Funktionen. Detailiertere Informationen können direkt dem Quellcode entnommen werden.

```
MODULE Benchmark;

  (*
    Purpose:  Interactive DM Variant - last generation using DM 3.1.4

    Remark:   The following language systems are used:

                - Lilith (Wirth, 1981)
                - MacMETH (Wirth et al., 1992)
                - P1 Modula2
                - epc Modula-2 for Solaris

    Cited References
    ----------------

    Wirth, N., 1981.  The personal computer LILITH. Internal
        Report 40, Institut fr Informatik, ETH, Zrich, 1-68 pp.

    Wirth, N., Gutknecht, J., Heiz, W., Schr, H., Seiler, H.,
        Vetterli, C. & Fischlin, A., 1992.  MacMETH. A fast Modula-2
        language system for the Apple Macintosh.  User Manual.  4th,
        completely revised ed.  User Manual Department of Computer
        Sciences (ETH), Zrich, Switzerland, 116 pp.

  *)

 (* ------------------------------------------------------------------------
    NOTE: This module contains version dependent code, since
    platforms provide different mechanisms to implement some
    of required functions. The code is constructed from the master
    implementation module by editing (commenting and
    uncommenting) locations marked with following conditional
    compilation comments:

    * VERSION_MacMETH          Version for DM on Macintosh
                               using the MacMETH compiler
      VERSION_P1               Version for P1 compiler on Mac

    * Generic master version in "SED:Dev: DevAux:Benchmarking:
    Benchmark Progs:Benchmark_a-o..q:Sources:DM-BatchDM:"
    ------------------------------------------------------------------------ *)

  (* IF VERSION_MacMETH *)
  (*$V-*)  (* integer arithmetic overflow check *)
  (*$R-*)  (* array index, subrange and pointer checks *)
  (* ENDIF VERSION_MacMETH *)
```

```
(*IF VERSION_P1 *) (*.
<* ASSIGN (OverflowCheck, FALSE) *>
<* ASSIGN (IndexCheck, FALSE) *>
<* ASSIGN (RangeCheck, FALSE) *>
<* ASSIGN (PointerCheck, FALSE) *>
<* ASSIGN (DivZeroCheck, FALSE) *>
<* ASSIGN (ComplexCheck, FALSE) *>
<* ASSIGN (Floatopt, TRUE) *>
<* ASSIGN (OptLevel, "2") *>
.*) (*ENDIF VERSION_P1 *)

(*Note: Since compiler options are usually implementation specific
make sure when porting this program module that compiler options are
also set properly at the beginning as well as the end of tests 'h'
and 'j'*)


(*
  a: empty REPEAT loop
  b: empty WHILE loop
  c: empty FOR loop
  d: CARDINAL arithmetic
  e: REAL arithmetic (basic operations)
  f: REAL transcendental functions
  g: CARDINAL vector by element assignment (array of single dimension)
  h: same as g but with index tests
  i: CARDINAL matrix by element assignment (matrix access)
  j: same as i but with index tests
  k: call of empty, parameterless procedure
  l: call of empty procedure with 4 parameters
  m: copying arrays (block moves)
  n: linear list traversal (pointer chaining)
  o: reading of file

  added for better interpretation of e and f results

  p: same as e but with LONGREAL
  q: same as f but with LONGREAL

  Note, tests f, o, and q actually test not only the compiler generated code,
  but depend on implementations of algorithms. These tests may easily show
  quite different results when importing from other library modules with
  other implementations. Moreover, there might be also several alternative
  implementations available for the very same library module. For instance
  a RAMSES release offers 4 variants of implementations of the library
  module DMMathLib (BatchDM, No SANE, No SANE20, SANE). Each may show a
  different performance.


  References:

  N. Wirth, 1981. The personal computer LILITH, IFI Bericht 40,
    Swiss Federal Institute of Technology Zurich, Zurich
    Switzerland, pp.66
  Loeffler, T.J., Lischke, H., Fischlin, A. & Ulrich, M., 1995.
    Benchmark experiments on workstations.  Systems Ecology
    Report No.  24, Institute of Terrestrial Ecology, Swiss
    Federal Institute of Technology ETH, Zurich, Switzerland,
    40 pp.

  Programming

    o Design
      Andreas Fischlin         20/01/2005

    o Implementation
      Andreas Fischlin         20/01/2005


  Swiss Federal Institute of Technology Zurich ETHZ
  Systems Ecology / Institute of Terrestrial Ecology
```

```
  Grabenstr. 3
  CH-8952 Schlieren/Zurich
  SWITZERLAND

  URLs:
    <mailto:RAMSES@env.ethz.ch>
    <http://www.ito.ethz.ch/SysEcol>
    <ftp://ftp.ito.ethz.ch/pub/mac/RAMSES>

  Implementation and Revisions:
  =============================

  Author  Date        Description of change
  ------  ----        ---------------------
  AF      20/01/2005  First implementation
  af      30/01/2005  Enhancing flexibility of program
                      to operate well and conveniently on
                      various platforms
  og&af   10/02/2005  V 1.0.2
                      - Fixing P1 incompatibilities
  af      10/02/2005  V 1.0.3
                      - Ensuring ReadWord (and WriteWord)
                      has to do the same effort regardless
                      of word size
  af      14/02/2005  V 1.0.4
                      - Adjustments for critical differences
                        in size of elementary data types such
                        as INTEGER or CARDINAL for better
                        comparability of tests (introducing
                        type Cardinal)
                      - Adding new critical LONGREAL tests
                        "p" and "q"
                      - Adding further comments to this
                        program (otherwise unchanged)

*)

FROM SYSTEM IMPORT WORD, BYTE;

(* DM Core *)
FROM DMStorage IMPORT Allocate, Deallocate;
FROM DMConversions IMPORT LongCardToString;
FROM DMWindIO IMPORT Write, WriteLn, WriteString;
FROM DMMaster IMPORT Read, BusyRead;
(* Convenient if available, since it supports interactive use,
otherwise omit following imports from DM core *)
FROM DMSystem IMPORT InstallTermProc, CurrentDMLevel,
  GetComputerName, FPUPresent, GetCPUName, GetFPUName,
  ComputerSystem, SystemVersion,
  RunsOnAMac, RunsOnAnIBMPC, RunsOnAUnixMachine,
  SUN, SUN3, SUNSparc, IBMPC, IBMAT, IBMPS2, IBMRisc6000;
IMPORT DMClock; (* Now *) FROM DMClock IMPORT NowInSeconds, Today;
FROM DMMessages IMPORT DoInform, Inform, Warn;
FROM DMLanguage IMPORT fileResBase;
FROM DMMenus IMPORT Menu, InstallMenu, Command, InstallCommand,
  AccessStatus, Marking, InstallAliasChar, InstallQuitCommand,
  CheckCommand, UncheckCommand, DisableMenu, EnableMenu;
FROM DMWindows IMPORT ReshowBackground;
FROM DMWindIO IMPORT EraseContent, SetPos,
  SetWindowFont, WindowFont, FontStyle;
FROM DMMaster IMPORT SetKeyboardHandlerMode,
  DialogMachineTask, ShowWaitSymbol, HideWaitSymbol,
  InitDialogMachine, DialogMachineIsRunning, RunDialogMachine,
  DialogMachineIsInBatchMode;

(* DM optional *)
FROM DMMathLib IMPORT Sin, Exp, Ln, Sqrt;
FROM DMLongMathLib IMPORT LongSin, LongExp, LongLn, LongSqrt;
FROM DMFiles IMPORT TextFile, neverOpenedFile;
FROM DMFiles IMPORT Response, Lookup, Close, Reset;
FROM DMFiles IMPORT ReadChar,
```

```
  WriteChar, WriteChars, WriteEOL, PutLongCard;

TYPE
  File = TextFile;


CONST
  versStr = "v1.0.4 - 4/Mar/2005";



(***********************************)
(******   Auxiliary Routines   ******)
(***********************************)


(*-----------------------*)
(*=====   Strings   =====*)
(*-----------------------*)

PROCEDURE Length(s: ARRAY OF CHAR): INTEGER;
  VAR i,hi: INTEGER;
BEGIN (* Length *)
  i := 0; hi := HIGH(s);
  WHILE (i<=hi) AND (s[i]<>0C) DO INC(i) END(*WHILE*);
  RETURN i
END Length;


CONST
  stopAt0C = -1;

PROCEDURE CopyString (from: ARRAY OF CHAR;   i1,nrChs: INTEGER;
         VAR to: ARRAY OF CHAR; VAR i2: INTEGER);
  (* VAR for from only for speed *)
  (* IF nrChs<0 THEN end at from[i] = 0C or HIGH(from) ELSE after
  copying nrChs chars *)
  VAR n1,n2,nn: INTEGER;
BEGIN
  n1 := HIGH(from); n2 := HIGH(to);
  IF nrChs<=stopAt0C THEN
    WHILE (i1<=n1) AND (from[i1]<>0C) AND (i2<=n2) DO
      to[i2] := from[i1]; INC(i1); INC(i2);
    END(*WHILE*);
  ELSE
    IF i2+nrChs-1<n2 THEN nn := i2+nrChs-1 ELSE nn := n2 END;
    WHILE (i2<=nn) AND (i1<=n1) DO
      to[i2] := from[i1]; INC(i1); INC(i2);
    END(*WHILE*);
  END(*IF*);
  IF i2<=n2 THEN to[i2] := 0C END;
END CopyString;

PROCEDURE AssignString (s: ARRAY OF CHAR; VAR d: ARRAY OF CHAR);
  VAR i: INTEGER;
BEGIN (* AssignString *)
  i:= 0; CopyString(s,0,stopAt0C,d,i);
END AssignString;

PROCEDURE Concatenate (s1,s2: ARRAY OF CHAR; VAR d: ARRAY OF CHAR);
  VAR i: INTEGER;
BEGIN (* Concatenate *)
  i:= 0;
  CopyString(s1,0,stopAt0C,d,i);
  CopyString(s2,0,stopAt0C,d,i);
END Concatenate;

PROCEDURE Append (VAR d: ARRAY OF CHAR; e: ARRAY OF CHAR);
  VAR i: INTEGER;
BEGIN (* Append *)
  i := Length(d);
```

```
  CopyString(e,0,stopAt0C,d,i);
END Append;


(*------------------------*)
(*=====   File I/O   =====*)
(*------------------------*)

CONST
  LilithWordSize = 2;

PROCEDURE ReadWord (VAR f: TextFile; VAR w: WORD); (* not available from DM *)
  VAR i: INTEGER; ch: CHAR;
BEGIN (* ReadWord *)
  FOR i:= 1 TO LilithWordSize DO
    ReadChar(f,ch);
  END(*FOR*);
END ReadWord;


PROCEDURE WriteWord (VAR f: TextFile; w: WORD); (* ignores w! *)
  CONST zero = "0"; one = "L";
  VAR i: INTEGER;
BEGIN (* WriteWord *)
  FOR i:= 1 TO LilithWordSize DO
    IF ODD(i) THEN WriteChar(f,zero) ELSE WriteChar(f,one) END;
  END(*FOR*);
END WriteWord;

PROCEDURE LongCardAsString (x: LONGCARD; VAR s: ARRAY OF CHAR);
BEGIN (* LongCardAsString *)
  LongCardToString(x,s,0);
END LongCardAsString;


(*----------------------*)
(*=====   System   =====*)
(*----------------------*)

PROCEDURE ReportFileError (VAR(*speed-up*) f: File; proc: ARRAY OF CHAR);
BEGIN (* ReportFileError *)
  DoInform(fileResBase+ORD(f.res), "Benchmark",proc,f.filename);
END ReportFileError;


PROCEDURE EnableKeyboardListening;
  CONST maxPrio = 0;
BEGIN (* EnableKeyboardListening *)
  SetKeyboardHandlerMode(TRUE,maxPrio);
END EnableKeyboardListening;

PROCEDURE DisableKeyboardListening;
  CONST maxPrio = 0;
BEGIN (* DisableKeyboardListening *)
  SetKeyboardHandlerMode(FALSE,maxPrio);
END DisableKeyboardListening;



PROCEDURE Now (): LONGCARD;
BEGIN (* Now *)
  (* return 0 if clock not available *)
  RETURN NowInSeconds()
END Now;

PROCEDURE UpdateSystem;
BEGIN (* UpdateSystem *)
  DialogMachineTask
END UpdateSystem;
```

```
(*******************************************************)
(******   Global Benchmark Parameters and Objects   ######*)
(*******************************************************)

(* The following parameters can be modified according to needs *)

CONST
  commandToolVariant = FALSE; (* if TRUE requires code adjustments *)
  defaultBatchMode = FALSE; (* if TRUE requires availability of clock
                               and runs entire program as a batch program
                               only! *)
  repetitions = 3; (* in batch mode repeat every test repetitions times *)
  defaultTestTime = 60; (* some previous benchmarking used 60 (Wirth, 1981),
                           others 100 (Loeffler et al., 1995) *)
  scale = 50;  (* Allows for adjusting for increased speed of machines
                  1 - Lilith; ~100 - PowerMac G4 1 GHz
                  Must not be > MAX(CARDINAL) DIV testTime
                  (e.g. for MacMETH 655) and reduces accuracy
                  if chosen too large.  *)
  lscale = LONGCARD(scale);
  foutFName = "Benchmark_Results.txt"; (* File in which to store
                                          benchmark results *)
  readFileName =
  "BenchmarkReadFile.txt"; (*
                             Either provide this file before
                             running this program or it will be
                             created.
                           *)



CONST
  noOfReads = 5000;

TYPE
  NodePtr = POINTER TO Node;
  Cardinal = RECORD b: ARRAY [1..LilithWordSize] OF BYTE END(*RECORD*);
  Node = RECORD x,y: Cardinal; next: NodePtr END;

VAR A,B,C: ARRAY [0..255] OF Cardinal;
  M: ARRAY [0..99],[0..99] OF Cardinal;
  m: CARDINAL; head: NodePtr;
  ch: CHAR;
  n: CARDINAL; nlong: LONGCARD;
  f: File;
  q: NodePtr;
  v73: Cardinal;

VAR
  testTime: LONGCARD;
  testTimeStr: ARRAY [0..15] OF CHAR;

  fout: File;
  foutReady: BOOLEAN;

  loadLevel: CARDINAL;

VAR
  fileM: Menu;
  toggleCmd, bmtCmd: Command;


CONST
  ESC = 33C;
```

```
(***********************************)
(*#####   Batch Mode Support   #####*)
(***********************************)

CONST
  bufHi = 2047;

VAR
  batchMode: BOOLEAN;
  notInterruptable: BOOLEAN;
  testStartedAt: LONGCARD;
  batchBuf: ARRAY [0..bufHi] OF CHAR;
  bbPos: INTEGER;


PROCEDURE InitBatchBuf;
  VAR i,j,k,a,o,q: INTEGER;
BEGIN (* InitBatchBuf *)
  k := 0; a := ORD("a"); o := ORD("o"); q := ORD("q");
  FOR i:= a TO q DO
    FOR j:= 1 TO repetitions DO
      batchBuf[k] := CHR(i); INC(k);
    END(*FOR*);
  END(*FOR*);
  IF k<=bufHi THEN batchBuf[k] := 0C END;
END InitBatchBuf;


PROCEDURE GetTestKind (VAR kind: CHAR);
BEGIN (* GetTestKind *)
  IF NOT batchMode THEN
    Read(ch);
  ELSE
    IF (bbPos<=bufHi) AND (batchBuf[bbPos]<>0C) THEN
      testStartedAt := Now();
      kind := batchBuf[bbPos]
    ELSE
      kind := ESC; (* for quit *)
    END(*IF*);
    INC(bbPos);
  END(*IF*);
END GetTestKind;


PROCEDURE CheckIfFinished (VAR stopCh: CHAR);
BEGIN (* CheckIfFinished *)
  BusyRead(stopCh);
  IF batchMode THEN
    IF (stopCh=".") OR (stopCh=ESC) THEN
      bbPos := bufHi+1;
      stopCh := ESC; (* for quit *)
    ELSIF (Now() - testStartedAt) >= testTime THEN
      stopCh := ESC; (* for quit *)
    ELSE
      stopCh := 0C; (* for continue *)
    END(*IF*);
  END(*IF*);
END CheckIfFinished;




(*****************************************************)
(*#####   Auxiliary Routines for Output of Results   #####*)
(*****************************************************)

PROCEDURE UserHasQuit(): BOOLEAN;
BEGIN (* UserHasQuit *)
```

```
  IF notInterruptable THEN
    RETURN FALSE
  ELSE
    RETURN NOT DialogMachineIsRunning()
  END;
END UserHasQuit;


(*--------------------------------*)
(*=====   Messages/Dialogs   =====*)
(*--------------------------------*)

CONST
  warn   = TRUE  (*offerDebugging*);
  inform = FALSE (*offerDebugging*);

PROCEDURE GiveMessage (s1,s2,s3: ARRAY OF CHAR; offerDebugging: BOOLEAN);
BEGIN (* GiveMessage *)
  IF NOT defaultBatchMode THEN
    IF offerDebugging THEN
      Warn(s1,s2,s3);
    ELSE
      Inform(s1,s2,s3);
    END(*IF*);
  ELSE
    WriteLn;
    IF offerDebugging THEN
      WriteString("Warning: ");
    ELSE
      WriteString("Note: ");
    END(*IF*);
    WriteString(s1); WriteLn;
    WriteString(s2); WriteLn;
    WriteString(s3); WriteLn;
  END(*IF*);
END GiveMessage;


(*--------------------------------------*)
(*=====   Window/Terminal output   =====*)
(*--------------------------------------*)

CONST
  monoFont = Monaco;

PROCEDURE PresentOutputCanvas; (* readies and clears any output media
                                  such as a window *)
BEGIN (* PresentOutputCanvas *)
  ReshowBackground; (* use background for output of results *)
  SetWindowFont(monoFont,9,FontStyle{});
  EraseContent; SetPos(1,1);
END PresentOutputCanvas;

PROCEDURE WriteCardinal (x: LONGCARD); (* writes every third digit a "'" *)
  VAR i,n,m: INTEGER; xs: ARRAY [0..31] OF CHAR;
BEGIN (* WriteCardinal *)
  LongCardAsString(x,xs); m := HIGH(xs);
  i:= 0; n := Length(xs) MOD 3;
  WHILE (i<=m) AND (xs[i]<>0C) DO
    Write(xs[i]);
    IF (i+1>=n) AND ((i<m) AND (xs[i+1]<>0C)) AND ((i+1-n) MOD 3 = 0) THEN Write("'") END;
    INC(i);
  END(*WHILE*);
END WriteCardinal;


VAR
  progCounter: CARDINAL;

PROCEDURE ShowProgress;
  CONST BS=10C;
```

```
BEGIN (* ShowProgress *)
  IF commandToolVariant THEN
    INC(progCounter);
    CASE  progCounter MOD 8 OF
    | 0,4:  Write("|");
    | 1,5:  Write("/");
    | 2,6:  Write("-");
    | 3,7:  Write("\");
    END(*CASE*);
    Write(BS);
  ELSE
    ShowWaitSymbol; (* omittable *)
  END(*IF*);
END ShowProgress;

PROCEDURE StopShowProgress;
BEGIN (* StopShowProgress *)
  IF commandToolVariant THEN
    progCounter := 0;
    Write(" ");
  ELSE
    HideWaitSymbol; (* omittable *)
  END(*IF*);
END StopShowProgress;


(*--------------------------*)
(*======   File output   =====*)
(*--------------------------*)

PROCEDURE PutCh (VAR f: File; ch: CHAR);
BEGIN (* PutCh *)
  WriteChar(f,ch);
END PutCh;

PROCEDURE PutChs (VAR f: File; s: ARRAY OF CHAR);
BEGIN (* PutChs *)
  WriteChars(f,s);
END PutChs;

PROCEDURE PutLCard (VAR f: File; lc: LONGCARD);
BEGIN (* PutLCard *)
  PutLongCard(f,lc,0);
END PutLCard;

PROCEDURE PutEOL (VAR f: File);
BEGIN (* PutEOL *)
  WriteEOL(f)
END PutEOL;

PROCEDURE ReadyOutputFile;
  CONST TAB = 11C;
  PROCEDURE DocuSystem;
    VAR cname, cpuname, fpuname: ARRAY [0..255] OF CHAR;
      withFPU: BOOLEAN; compSysNo: INTEGER; sysV: REAL;
  BEGIN (* DocuSystem *)
    GetComputerName(cname); (* assign accordingly if unavailable *)
    GetCPUName(cpuname);     (* assign accordingly if unavailable *)
    withFPU := FPUPresent(); (* assign accordingly if unavailable *)
    GetFPUName(fpuname);     (* assign accordingly if unavailable *)
    compSysNo := ComputerSystem(); (* assign 0 if unavailable *)
    sysV := SystemVersion();       (* assign 0.0 if unavailable *)

    PutChs(fout,"Benchmark Tests using "); PutChs(fout,versStr); PutChs(fout," on ");
    IF RunsOnAMac() THEN
      PutChs(fout,"a Macintosh");
    ELSIF RunsOnAnIBMPC() THEN
      PutChs(fout,"an IBM PC");
    ELSIF RunsOnAUnixMachine() THEN
      PutChs(fout,"a Unix machine");
    ELSE
```

```
      PutChs(fout,"Unknown computer");
    END(*IF*);
    PutChs(fout,":"); PutEOL(fout);
    IF sysV<>0.0 THEN
      PutChs(fout,"Operating system: ");
      PutLCard(fout,TRUNC(sysV)); PutCh(fout,".");
      PutLCard(fout,TRUNC(100.0*(sysV - FLOAT(TRUNC(sysV)))));
      PutChs(fout," ");
    END(*IF*);
    PutChs(fout,cname); PutChs(fout,"   ");
    PutChs(fout,cpuname); PutChs(fout,"   ");
    PutEOL(fout);
    IF compSysNo <> 0 THEN
      PutChs(fout,"ComputerSystem# = "); PutLCard(fout,compSysNo);
      IF RunsOnAMac() THEN
        PutChs(fout," (see Mac specs at: http://www.info.apple.com/support/applespec.html)");
      ELSE
        CASE compSysNo OF
        | SUN, SUN3, SUNSparc:  WriteString(" Sun");
        | IBMPC, IBMAT, IBMPS2:  WriteString(" IBM PC");
        | IBMRisc6000:  WriteString(" IBM RISC");
        ELSE
          (* do nothing *)
        END(*CASE*);
      END(*IF*);
      PutEOL(fout);
    END(*IF*);
    IF withFPU THEN
      PutChs(fout,"FPU present");
      PutChs(fout," ("); PutChs(fout,fpuname); PutCh(fout,")");
    ELSE
      PutChs(fout,"No FPU present")
    END(*IF*);
    PutEOL(fout);
  END DocuSystem;
  PROCEDURE DocuDateTime;
    VAR year, month, day, dayOfWeek: INTEGER;
      hour, minute, second: INTEGER;
  BEGIN (* DocuDateTime *)
    Today(year, month, day, dayOfWeek);
    DMClock.Now(hour, minute, second);
    PutLCard(fout,day); PutCh(fout,"/"); PutLCard(fout,month); PutCh(fout,"/"); PutLCard(fout,year);
    PutChs(fout,"    ");
    PutLCard(fout,hour); PutCh(fout,"h"); PutLCard(fout,minute); PutCh(fout,":"); PutLCard(fout,second);
    PutEOL(fout);
  END DocuDateTime;
  PROCEDURE DocuTestTime;
  BEGIN (* DocuTestTime *)
    PutChs(fout,"Test time: "); PutChs(fout,testTimeStr); PutChs(fout," (seconds)");
    PutEOL(fout);
  END DocuTestTime;
BEGIN (* ReadyOutputFile *)
  foutReady := FALSE;
  AssignString(foutFName,fout.filename);
  Lookup(fout,fout.filename,TRUE(*new*));
  IF (fout.res=done) THEN
    foutReady := TRUE;
    DocuSystem; DocuDateTime; DocuTestTime;
    PutEOL(fout); PutEOL(fout);
    PutChs(fout,"Test"); PutCh(fout,TAB);
    PutChs(fout,"Iterations"); PutCh(fout,TAB);
    PutChs(fout,"Scale"); PutCh(fout,TAB);
    PutChs(fout,"n/tsec"); PutCh(fout,TAB);
    PutChs(fout,"tsec"); PutCh(fout,TAB);
    PutChs(fout,"n/"); PutChs(fout,testTimeStr);
    PutEOL(fout);
  ELSE (* an error occurred *)
    ReportFileError(fout,"RunBenchmark");
  END(*IF*);
END ReadyOutputFile;
```

```
PROCEDURE DocuTest1stPart (kind: CHAR; nlong,lscale,nCounted: LONGCARD);
  CONST TAB = 11C;
BEGIN (* DocuTest1stPart *)
  IF NOT foutReady THEN RETURN END;
  PutCh(fout,kind); PutCh(fout,TAB);
  PutLCard(fout,nlong); PutCh(fout,TAB);
  PutLCard(fout,lscale); PutCh(fout,TAB);
  PutLCard(fout,nCounted); PutCh(fout,TAB);
END DocuTest1stPart;

PROCEDURE DocuTest2ndPart (elapsedTime,nPerTestTime: LONGCARD);
  CONST TAB = 11C;
BEGIN (* DocuTest2ndPart *)
  IF NOT foutReady THEN RETURN END;
  PutLCard(fout,elapsedTime); PutCh(fout,TAB);
  PutLCard(fout,nPerTestTime);
  PutEOL(fout);
END DocuTest2ndPart;

PROCEDURE FinishOutputFile;
BEGIN (* FinishOutputFile *)
  IF NOT foutReady THEN RETURN END;
  PutEOL(fout); PutEOL(fout);
  PutChs(fout,"References:"); PutEOL(fout);
  PutChs(fout,"  N. Wirth, 1981. The personal computer LILITH, IFI Bericht 40,"); PutEOL(fout);
  PutChs(fout,"    Swiss Federal Institute of Technology Zurich, Zurich"); PutEOL(fout);
  PutChs(fout,"    Switzerland, pp.66"); PutEOL(fout);
  PutChs(fout,"  Loeffler, T.J., Lischke, H., Fischlin, A. & Ulrich, M., 1995."); PutEOL(fout);
  PutChs(fout,"    Benchmark experiments on workstations.  Systems Ecology"); PutEOL(fout);
  PutChs(fout,"    Report No.  24, Institute of Terrestrial Ecology, Swiss"); PutEOL(fout);
  PutChs(fout,"    Federal Institute of Technology ETH, Zurich, Switzerland, 40 pp."); PutEOL(fout);
  Close(fout);
  foutReady := FALSE;
END FinishOutputFile;



(*********************************)
(*#####   Benchmark Tests   #####*)
(*********************************)

PROCEDURE Test(ch: CHAR);
  VAR i,j,k: CARDINAL;
    r0, r1, r2: REAL; lr0, lr1, lr2: LONGREAL; p: NodePtr;

  PROCEDURE P;
  BEGIN
  END P;

  PROCEDURE Q(x,y,z,w: CARDINAL);
  BEGIN
  END Q;

BEGIN (*Test*)
  CASE ch OF
    "a":  k := 20000;
          REPEAT
            k := k-1
          UNTIL k = 0 |

    "b":  i := 20000;
          WHILE i > 0 DO
            i := i-1
          END |

    "c":  FOR i := 1 TO 20000 DO
          END |

    "d":  j := 0; k := 10000;
          REPEAT
            k := k-1;  j := j+1; i := (k*3) DIV (j*5)
```

```
      UNTIL k = 0 |

"e":  k := 5000; r1 := 7.28;  r2 := 34.8;
      REPEAT
        k := k-1;  r0 := (r1*r2) / (r1+r2)
      UNTIL k = 0 |

"f":  k := 500;
      REPEAT
        r0 := Sin(0.7); r1 := Exp(2.0);
        r0 := Ln(10.0); r1 := Sqrt(18.0);
        k := k-1
      UNTIL k = 0 |

"g":  k := 20000;  i := 0;  B[0] := v73;
      REPEAT
        A[i] := B[i];  B[i] := A[i];  k := k-1
      UNTIL k = 0 |

"h":
      (* IF VERSION_MacMETH *)
      (*$V+*)  (* integer arithmetic overflow check *)
      (*$R+*)  (* array index, subrange and pointer checks *)
      (* ENDIF VERSION_MacMETH *)

      (*IF VERSION_P1 *) (*.
      <* PUSH *>
      <* ASSIGN (OverflowCheck, TRUE) *>
      <* ASSIGN (IndexCheck, TRUE) *>
      <* ASSIGN (RangeCheck, TRUE) *>
      <* ASSIGN (PointerCheck, TRUE) *>
      .*) (*ENDIF VERSION_P1 *)

      k := 20000;  i := 0;  B[0] := v73;
      REPEAT
        A[i] := B[i];  B[i] := A[i];  k := k-1
      UNTIL k = 0

      (* IF VERSION_MacMETH *)
      (*$V=*) (* restore previous (global) settings *)
      (*$R=*) (* restore previous (global) settings *)
      (* ENDIF VERSION_MacMETH *)

      (*IF VERSION_P1 *) (*.
      <* POP *>
      .*) (*ENDIF VERSION_P1 *)

      |

"i":  FOR i := 0 TO 99 DO
        FOR j := 0 TO 99 DO
          M[i,j] := M[j,i]
        END
      END |

"j":
      (* IF VERSION_MacMETH *)
      (*$V+*)  (* integer arithmetic overflow check *)
      (*$R+*)  (* array index, subrange and pointer checks *)
      (* ENDIF VERSION_MacMETH *)

      (*IF VERSION_P1 *) (*.
      <* PUSH *>
      <* ASSIGN (OverflowCheck, TRUE) *>
      <* ASSIGN (IndexCheck, TRUE) *>
      <* ASSIGN (RangeCheck, TRUE) *>
      <* ASSIGN (PointerCheck, TRUE) *>
      .*) (*ENDIF VERSION_P1 *)

      FOR i := 0 TO 99 DO
        FOR j := 0 TO 99 DO
```

```
            M[i,j] := M[j,i]
          END
        END

        (* IF VERSION_MacMETH *)
        (*$V=*) (* restore previous (global) settings *)
        (*$R=*) (* restore previous (global) settings *)
        (* ENDIF VERSION_MacMETH *)

        (*IF VERSION_P1 *) (*.
        <* POP *>
        .*) (*ENDIF VERSION_P1 *)

        |

    "k":  k := 20000;
        REPEAT
          P;  k := k-1;
        UNTIL k = 0 |

    "l":  k := 20000;
        i := 1; j := 2; m := 3;
        REPEAT
          Q(i,j,k,m);  k := k-1;
        UNTIL k = 0 |

    "m":  k := 500;
        REPEAT
          k := k-1;  A := B;  B := C;  C := A
        UNTIL k = 0 |

    "n":  k := 500;
        REPEAT p := head;
          REPEAT p := p^.next UNTIL p = NIL;
          k := k-1
        UNTIL k = 0 |

    "o":  k := noOfReads;
        REPEAT
          k := k-1; ReadWord(f,i)
        UNTIL k = 0;
        Reset(f) |

    (* new tests added to be able to compare "e" and "f"
    with LONGREAL results *)

    "p":  k := 5000; lr1 := 7.28;  lr2 := 34.8;
        REPEAT
          k := k-1;  lr0 := (lr1*lr2) / (lr1+lr2)
        UNTIL k = 0 |

    "q":  k := 500;
        REPEAT
          lr0 := LongSin(0.7); lr1 := LongExp(2.0);
          lr0 := LongLn(10.0); lr1 := LongSqrt(18.0);
          k := k-1
        UNTIL k = 0

  END(*CASE*)
END Test;



(****************************************)
(*#####   Benchmark Preconditions   #####*)
(****************************************)

PROCEDURE ReadFilePresent(): BOOLEAN;
  VAR present: BOOLEAN; k: INTEGER; dummyWord: WORD;
BEGIN (* ReadFilePresent *)
  AssignString(readFileName,f.filename);
```

```
  Lookup(f,f.filename,FALSE(*new*));
  present := f.res=done;
  IF NOT present AND
    ((f.res=filenotfound) OR (f.res=volnotfound) OR (f.res=unknownfile))
  THEN
    Lookup(f,f.filename,TRUE(*new*));
    IF (f.res=done) THEN
      dummyWord := WORD(0);
      FOR k:= 0 TO 2*noOfReads DO
        WriteWord(f,dummyWord);
      END(*FOR*);
      Close(f);
      (* test its accessibility once more *)
      AssignString(readFileName,f.filename);
      Lookup(f,f.filename,FALSE(*new*));
      present := f.res=done;
    ELSE (* an error occurred *)
      ReportFileError(f,"RunBenchmark");
      RETURN FALSE (* give up, since can't create *)
    END(*IF*);
  END(*IF*);
  IF present THEN
    (* test its readability *)
    Test("o");
    Close(f);
    present := f.res=done
  ELSE
    ReportFileError(f,"RunBenchmark");
    Close(f);
  END(*IF*);
  RETURN present;
END ReadFilePresent;

PROCEDURE ScaleOk(): BOOLEAN;
  VAR maxScale: LONGCARD; msg: ARRAY [0..255] OF CHAR;
BEGIN (* ScaleOk *)
  maxScale := LONGCARD(MAX(CARDINAL));
  maxScale := maxScale DIV testTime;
  IF lscale <= maxScale THEN
    RETURN TRUE
  ELSE
    Concatenate("Scale larger than MAX(CARDINAL) DIV ",testTimeStr,msg);
    Append(msg,"!"); GiveMessage(msg,"","",warn);
    RETURN FALSE
  END(*IF*);
END ScaleOk;

PROCEDURE ClockAvailable (): BOOLEAN;
BEGIN (* ClockAvailable *)
  RETURN Now() > testTime;
END ClockAvailable;



(*********************************************)
(*#####   Execution of Benchmark Tests   #####*)
(*********************************************)

PROCEDURE Prepare;
BEGIN (* Prepare *)
  DisableMenu(fileM); (* disable also quit command! *)
  Lookup(f,readFileName,FALSE);
  head := NIL;  n := 100;
  REPEAT q := head;
    Allocate(head,SIZE(Node)); head^.next := q; n := n-1
  UNTIL n = 0;
  ReadyOutputFile;
  bbPos := 0;
END Prepare;
```

```
PROCEDURE Cleanup;
  VAR q: NodePtr;
BEGIN (* Cleanup *)
  EnableMenu(fileM);
  IF CurrentDMLevel()=loadLevel THEN
    Close(f);
    WHILE head<>NIL DO
      q := head; head := head^.next; Deallocate(q);
    END(*WHILE*);
    FinishOutputFile;
  END(*IF*);
END Cleanup;


PROCEDURE RunBenchmark;
  VAR i: CARDINAL;
    stopCh: CHAR;
    msg: ARRAY [0..255] OF CHAR;
    startTime, stopTime: LONGCARD;

  PROCEDURE ReportResults;
    VAR nCounted, nPerTestTime,elapsedTime,zero: LONGCARD;
  BEGIN (* ReportResults *)
    WriteLn;
    WriteString("n = "); nlong := n; nCounted := nlong*lscale;
    WriteCardinal(nCounted);
    WriteString(" (="); WriteCardinal(n);
    WriteString("*"); WriteCardinal(scale); WriteString(") ");
    DocuTest1stPart(ch,nlong,lscale,nCounted);
    IF ClockAvailable() THEN
      elapsedTime := stopTime-startTime;
      WriteCardinal(elapsedTime); WriteString(" sec"); WriteLn;
      Concatenate(testTimeStr," sec corrected n = ",msg); WriteString(msg);
      zero := 0;
      IF elapsedTime>zero THEN
        IF MAX(LONGCARD) DIV (nlong*lscale) >= testTime THEN
          nPerTestTime := (nlong*lscale*testTime) DIV elapsedTime;
        ELSE
          GiveMessage("Test time too long, you may need to reduce it to avoid CARDINAL overflow","","",warn);
          nPerTestTime := ((nlong*lscale) DIV elapsedTime)*testTime;
        END(*IF*);
      ELSE
        nPerTestTime := 0;
      END(*IF*);
      WriteCardinal(nPerTestTime);
      DocuTest2ndPart(elapsedTime,nPerTestTime);
    END(*IF*);
  END ReportResults;

BEGIN (* RunBenchmark *)
  IF NOT ReadFilePresent() THEN RETURN END;
  IF NOT ScaleOk() THEN RETURN END;

  Prepare;

  IF NOT batchMode THEN
    AssignString("Entering benchmark interactive mode. ",msg);
  ELSE
    AssignString("Entering benchmarking in batch mode. ",msg);
  END(*IF*);
  Append(msg,"Usage: ");
  IF NOT batchMode THEN
    Append(msg,"Type a..q for a test, any other key to stop. ");
    Append(msg,"Run each test for ~100 seconds. ");
  ELSE
    IF RunsOnAUnixMachine() THEN
      Append(msg,"Ctrl-C interrupts. ");
    ELSE
      Append(msg,"ESC interrupts. ");
    END(*IF*);
  END(*IF*);
```

```
  Append(msg,"Results written to file '"); Append(msg,foutFName); Append(msg,"'. ");
  GiveMessage(msg,"","",inform);
  PresentOutputCanvas;
  StopShowProgress;
  EnableKeyboardListening;
  Write(">"); GetTestKind(ch);
  WHILE ("a" <= ch) & (ch < "r") DO
    Write(ch); WriteLn; n :=0;
    ShowProgress;
    startTime := Now();
    REPEAT
      IF n=MAX(CARDINAL) THEN GiveMessage("n overflowed","results invalid","",warn) END;
      n := n+1;
      FOR i:= 1 TO scale DO
        IF i MOD 100 = 0 THEN ShowProgress END;
        Test(ch);
      END(*FOR*);
      IF (n MOD 50) = 0 THEN WriteLn END;
      Write("."); CheckIfFinished(stopCh); UpdateSystem;
    UNTIL (stopCh<>0C) OR UserHasQuit();
    stopTime := Now();
    StopShowProgress;
    ReportResults;
    WriteLn; Write(">"); GetTestKind(ch)
  END(*WHILE*);
  DisableKeyboardListening;
  IF NOT batchMode THEN
    GiveMessage("Leaving benchmark interactive mode","","",inform);
  ELSE
    GiveMessage("Leaving benchmarking in batch mode","","",inform);
  END(*IF*);

  Cleanup;
END RunBenchmark;



(*********************************)
(*#####   Menu Management   #####*)
(*********************************)

PROCEDURE ToggleBatchMode;
BEGIN (* ToggleBatchMode *)
  batchMode := NOT batchMode;
  IF batchMode THEN
    CheckCommand(fileM,toggleCmd);
  ELSE
    UncheckCommand(fileM,toggleCmd);
  END(*IF*);
END ToggleBatchMode;

PROCEDURE Quitting(VAR quit: BOOLEAN);
BEGIN (* Quitting *)
  ch := ESC; quit := TRUE;
END Quitting;

PROCEDURE InstallMenus;
BEGIN (* InstallMenus *)
  InstallMenu(fileM,"File",enabled);
  InstallCommand(fileM,toggleCmd,"Batch Mode",ToggleBatchMode,
                 enabled,unchecked);
  IF batchMode THEN
    CheckCommand(fileM,toggleCmd);
  ELSE
    UncheckCommand(fileM,toggleCmd);
  END(*IF*);
  InstallCommand(fileM,bmtCmd,"Benchmark Test",RunBenchmark,
                 enabled,unchecked);
  InstallAliasChar(fileM,bmtCmd,"T");
  InstallQuitCommand("Quit",Quitting,"Q");
END InstallMenus;
```

```
  (*********************************)
  (*#####   Module Management   ######*)
  (*********************************)

  PROCEDURE AtTermBenchmark;
  BEGIN (* AtTermBenchmark *)
    Cleanup;
  END AtTermBenchmark;

  PROCEDURE AtInitBenchmark;
    VAR done: BOOLEAN;
  BEGIN (* AtInitBenchmark *)
    batchMode := defaultBatchMode;
    (* on a Unix machine or in DM batch mode force batch mode always *)
    IF RunsOnAUnixMachine() OR DialogMachineIsInBatchMode() THEN batchMode := TRUE END;
    IF batchMode AND NOT ClockAvailable() THEN batchMode := FALSE END;
    notInterruptable := RunsOnAUnixMachine() OR DialogMachineIsInBatchMode()
                        OR (defaultBatchMode AND batchMode);
    InitBatchBuf; bbPos := 0;
    foutReady := FALSE;
    head := NIL;
    m := 0;
    v73.b[1] := BYTE(CHR(0));
    v73.b[2] := BYTE(CHR(73));
    testTime := defaultTestTime;
    (* modify or omit following depending on availability *)
    LongCardAsString(testTime,testTimeStr); (* inites testTimeStr by testTime *)
    f := neverOpenedFile; (* inites f *)
    fout := neverOpenedFile; (* inites fout *)
    InstallMenus;
    loadLevel := CurrentDMLevel(); (* inite 0 if CurrentDMLevel unavailable *)
    InstallTermProc(AtTermBenchmark,done); (* ensures proper cleanup in all cases *)
  END AtInitBenchmark;


BEGIN (*BenchMark*)
  AtInitBenchmark;
  IF batchMode THEN
    InitDialogMachine;
    RunBenchmark;
  ELSE
    RunDialogMachine;
  END(*IF*);
END Benchmark.
```

D Testprogramme

D.1 Eigenschaften der Sprachdefinition – TestComp.mod

Dieses Programm testet die Sprachimplementation eines Kompilers, insbesondere die Typenkompatibilität.

```
MODULE TestComp;

  (*
    Purpose  This module was made only to test the behavior of Modula-2
             compilers at compile time. It tests in particular assignment
             compatibility and some issues of similar subtlety.

    Remarks  If a compiler succeeds in compiling the line of code
             the comment at the end of the line documents this fact
             by including the compiler's name.

             The following compilers were tested:
             - MACMETH 2.6.8  http://www.ito.umnw.ethz.ch/SysEcol/SimSoftware/RAMSES/MacMETH.html
             - P1 8.0         http://www.awiedemann.de/compiler/
             - epc 2.0.9.5    http://www.ito.umnw.ethz.ch/SysEcol/SimSoftware/RAMSES/em2_2.0.6.html
             - Stonybrook

    Implementation and Revisions:
    ============================

    Author  Date        Description of change
    ------  ----        ---------------------
    AF      14/10/2004  First implementation
    af      26/10/2004  Adjusted to latest special version P1 compiler
                        provided by A. Wiedemann on our request
  *)

  FROM SYSTEM IMPORT BYTE;
  FROM SYSTEM IMPORT VAL; (* MacMETH *)
  FROM SYSTEM IMPORT CAST; (* P1 *)

  TYPE
    MyType = POINTER TO MyRecordStructure;
    MyRecordStructure = RECORD x,y: INTEGER END;

  CONST
    x1 = LONGINT(1); (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
    x2 = CAST(LONGINT,1); (* VERSION: STONYBROOK, P1 *)
    x3 = VAL(LONGINT,1); (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
    y1 = LONGCARD(1); (* VERSION: MACMETH, STONYBROOK, EPC *)
    y2 = CAST(LONGCARD,1); (* VERSION: STONYBROOK, P1 *)
    y3 = VAL(LONGCARD,1); (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
    z1 = LONGREAL(0.1); (* VERSION: STONYBROOK *)
    z2 = CAST(LONGREAL,0.1); (* VERSION: STONYBROOK *)
    z3 = VAL(LONGREAL,0.1); (* VERSION: STONYBROOK, EPC, P1 *)
    p1 = MyType(NIL); (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
    p2 = CAST(MyType,NIL); (* VERSION: STONYBROOK, P1 *)
    p3 = VAL(MyType,NIL); (* VERSION: MACMETH, STONYBROOK *)

  VAR
    b:  BYTE; (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
    ch: CHAR; (* all *)
    i: INTEGER; (* all *)
    li: LONGINT; (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
    c:  CARDINAL; (* all *)
    lc: LONGCARD; (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
    r:  REAL; (* all *)
    lr: LONGREAL; (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
    bs: BITSET; (* all *)


BEGIN (* TestComp *)
```

```
  b := 0C; (* VERSION: MACMETH, STONYBROOK, EPC *)
  b := 0H; (* VERSION: STONYBROOK?, EPC *)
  b := BYTE(0C); (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  ch := 0C; (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  ch := CHR(1); (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  ch := CHAR(1); (* VERSION: MACMETH, STONYBROOK, EPC, P1-warning *)
  ch := VAL(CHAR,1); (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  ch := CAST(CHAR,1); (* VERSION: STONYBROOK, P1-warning *)
  i := c; (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  i := li; (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  i := lc; (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  li := i; (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  li := c; (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  li := lc; (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  c := i; (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  c := li; (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  c := lc; (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  lc := i; (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  lc := c; (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  lc := li; (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  r := FLOAT(i); (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  r := FLOAT(c); (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  r := FLOAT(li); (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
  r := FLOAT(lc); (* VERSION: STONYBROOK, EPC, P1 *)
  r := lr; (* VERSION: MACMETH, STONYBROOK, EPC *)
  lr := r; (* VERSION: MACMETH, STONYBROOK, EPC *)
  lr := LFLOAT(i); (* VERSION: STONYBROOK, EPC, P1 *)
  lr := LFLOAT(c); (* VERSION: STONYBROOK, EPC, P1 *)
  lr := LFLOAT(li); (* VERSION: STONYBROOK, EPC, P1 *)
  lr := LFLOAT(lc); (* VERSION: STONYBROOK, EPC, P1 *)
  lr := LFLOAT(lr); (* VERSION: STONYBROOK, EPC, P1 *)
  lr := LFLOAT(r); (* VERSION: STONYBROOK, EPC, P1 *)
  lr := FLOATD(i); (* VERSION: MACMETH *)
  lr := FLOATD(c); (* VERSION: MACMETH *)
  lr := FLOATD(li); (* VERSION: MACMETH *)
  lr := FLOATD(lc); (* VERSION: *)
  lr := FLOATD(lr); (* VERSION: *)
  lr := FLOATD(r); (* VERSION: *)
  bs := {}; (* VERSION: MACMETH, STONYBROOK, EPC *)
  bs := BITSET{}; (* VERSION: MACMETH, STONYBROOK, EPC, P1 *)
END TestComp.
```

D.2 Implemetation von SysDep – TestSysDep.mod

Dieses Modul testet alle, von dem Definitionsmodul SysDep.def exportierten Schnittstellen und überprüft damit die Implementation SysDep.mod auf deren Korrektheit.

```
MODULE TestSysDep;

  (*
    Implementation and Revisions:
    ============================

    Author  Date        Description of change
    ------  ----        ---------------------
    AF      30/10/2004  First implementation
  *)

  FROM SYSTEM IMPORT ADR, ADDRESS, BYTE;

  FROM SysDep IMPORT SDNullDevice, SDStdInput, SDStdOutput, SDStdError,
    SDEol, SDEol2, SDDirSeperator, SDM2PathEnvVarName,
    SDFile, SDFileResult,
    SDLookup, SDDelete, SDRename, SDClose, SDFlush, SDEOF,
    SDTruncateFile, SDFileRead, SDFileWrite,
    SDGetFileSize, SDGetFileDates, SDGetFilePos, SDSetFilePos,
    SDLastFileResult;
```

```
FROM SysDep IMPORT SDGetFileData;

FROM SysDep IMPORT SDValidPtr, SDAlloc, SDDealloc;

FROM SysDep IMPORT
  SDJan, SDFeb, SDMar, SDApr, SDMai, SDJun,
  SDJul, SDAug, SDSep, SDOct, SDNov, SDDec,
  SDSun, SDMon, SDTue, SDWed, SDThu, SDFri, SDSat,
  SDTime, SDDate,
  SDGetTime, SDTimeToDate, SDDateToTime;

FROM SysDep IMPORT
  SDDirectoryProc, SDCurWorkDirectory, SDCreateDir, SDDirInfo,
  SDDoForAllFilesInDirectory;

FROM SysDep IMPORT
  SDGetArgCount, SDGetArgument, SDGetEnvVar, SDGetEnvVarListElem;

FROM SysDep IMPORT
  SDExecute,
  SDLocalMsgID, SDLocalMsgProc,
  SDSetLocalMsgHandler, SDGetLocalMsgHandler, SDSendLocalMessage;

FROM SysDep IMPORT
  SDSetInitProc, SDSetTermProc, SDExit, SDExitMsg;

(* IF VERSION_ISO *)
FROM STextIO IMPORT WriteString, WriteChar, WriteLn;
FROM SWholeIO IMPORT WriteInt, WriteCard;
FROM SRealIO IMPORT WriteFixed;
(* ENDIF VERSION_ISO *)

(* IF VERSION_MacMETH *) (*.
FROM InOut IMPORT WriteString, WriteLn, WriteInt, WriteCard, WriteReal, Write;
PROCEDURE WriteChar(ch: CHAR); BEGIN Write(ch) END WriteChar;
PROCEDURE WriteFixed(r: REAL; n,fw: CARDINAL); BEGIN WriteReal(r,fw) END WriteFixed;
.*) (* ENDIF VERSION_MacMETH *)

(* IF VERSION_EPC *) (*.
FROM SimpleIO IMPORT WriteString, WriteLn, WriteInt, WriteCard, WriteChar;
FROM RealIO IMPORT WriteReal;
PROCEDURE WriteFixed(r: REAL; n,fw: CARDINAL); BEGIN WriteReal(r,fw,8) END WriteFixed;
.*) (* ENDIF VERSION_EPC *)


(*****************************)
(*#####   Aux Routines   #####*)
(*****************************)


(*---------------------------------------------------------------------------*)
(*=====   Strings, paths, and file names (cf. DMStrings, FileNameStrs)   ======*)
(*---------------------------------------------------------------------------*)

PROCEDURE Length(s: ARRAY OF CHAR): INTEGER;
  VAR i,hi: INTEGER;
BEGIN (* Length *)
  i := 0; hi := HIGH(s);
  WHILE (i<=hi) AND (s[i]<>0C) DO INC(i) END(*WHILE*);
  RETURN i
END Length;


CONST
  stopAt0C = -1;

PROCEDURE CopyString (VAR from: ARRAY OF CHAR;   i1,nrChs: INTEGER;
         VAR to: ARRAY OF CHAR; VAR i2: INTEGER);
  (* VAR for from only for speed *)
  (* IF nrChs<0 THEN end at from[i] = 0C or HIGH(from) ELSE after
  copying nrChs chars *)
```

```
    VAR n1,n2,nn: INTEGER;
  BEGIN
    n1 := HIGH(from); n2 := HIGH(to);
    IF nrChs<=stopAt0C THEN
      WHILE (i1<=n1) AND (from[i1]<>0C) AND (i2<=n2) DO
        to[i2] := from[i1]; INC(i1); INC(i2);
      END(*WHILE*);
    ELSE
      IF i2+nrChs-1<n2 THEN nn := i2+nrChs-1 ELSE nn := n2 END;
      WHILE (i2<=nn) AND (i1<=n1) DO
        to[i2] := from[i1]; INC(i1); INC(i2);
      END(*WHILE*);
    END(*IF*);
    IF i2<=n2 THEN to[i2] := 0C END;
  END CopyString;


  (*-----------------------------------*)
  (*=====   Conversion Routines   =====*)
  (*-----------------------------------*)

  PROCEDURE WriteHex (x: ARRAY OF BYTE);
    CONST upperCase = TRUE;
    VAR  hexDIGIT: ARRAY[0..16] OF CHAR;
      hstr: ARRAY [0..255] OF CHAR;

    PROCEDURE SetHexDigitsUpperCase( upperC: BOOLEAN );
    BEGIN
      IF upperC THEN  hexDIGIT:= '0123456789ABCDEF';
      ELSE            hexDIGIT:= '0123456789abcdef';
      END(*IF*);
    END SetHexDigitsUpperCase;

    TYPE
      ByteToChar = RECORD
CASE :BOOLEAN OF
    TRUE : ch : CHAR;
  | FALSE: b  : BYTE;
END;
      END;

    VAR
      i, ii, n, nb, nh, hihstr, tmp : INTEGER;
      byteToChar : ByteToChar;

  BEGIN
    SetHexDigitsUpperCase(upperCase);
    (* any testing for alignment or BYTE vs CHAR implementation problems done in Portab *)
    hihstr:= HIGH(hstr);
    nh:= (hihstr-1) DIV 2;
    nb:= HIGH(x);
    IF (nh < nb) THEN  n:= nh;
    ELSE               n:= nb;
    END(*IF*);
    ii:=  0;
    FOR i:= 0 TO n DO
      byteToChar.b := x[i];
      tmp := ORD( byteToChar.ch);
      hstr[ii]:= hexDIGIT[tmp DIV 16];  INC( ii );
      hstr[ii]:= hexDIGIT[tmp MOD 16];  INC( ii );
    END(*FOR*);
    IF (ii <= hihstr) THEN  hstr[ii]:= 0C;  END(*IF*);
    WriteString(hstr);
  END WriteHex;


  PROCEDURE CompletePathFileName (p,fn: ARRAY OF CHAR; VAR pfn: ARRAY OF CHAR);
    VAR i: INTEGER; dirSep: ARRAY [0..0] OF CHAR;
  BEGIN (* CompletePathFileName *)
    i := 0;
    CopyString(p,0,stopAt0C,pfn,i);
```

```
  IF (i>0) AND (pfn[i-1]<>SDDirSeperator) THEN
    dirSep[0] := SDDirSeperator;
    CopyString(dirSep,0,stopAt0C,pfn,i);
  END(*IF*);
  CopyString(fn,0,stopAt0C,pfn,i);
END CompletePathFileName;



PROCEDURE ReportFIOResult(op: ARRAY OF CHAR; VAR(*speed-up*) fn: ARRAY OF CHAR);
BEGIN (* ReportFIOResult *)
  WriteString("Operation "); WriteString(op);
  WriteString(" on file '"); WriteString(fn); WriteString("': ");
  CASE  SDLastFileResult() OF
  | SDDone:          WriteString("SDLastFileResult() = SDDone"         );
  | SDFileNotFound:  WriteString("SDLastFileResult() = SDFileNotFound" );
  | SDTooManyFiles:  WriteString("SDLastFileResult() = SDTooManyFiles" );
  | SDNoWriteAccess: WriteString("SDLastFileResult() = SDNoWriteAccess");
  | SDNoReadAccess:  WriteString("SDLastFileResult() = SDNoReadAccess" );
  | SDFileNameWrong: WriteString("SDLastFileResult() = SDFileNameWrong");
  | SDNotADirectory: WriteString("SDLastFileResult() = SDNotADirectory");
  | SDOtherResult:   WriteString("SDLastFileResult() = SDOtherResult"  );
  ELSE
    WriteString("SDLastFileResult() = "); WriteInt(ORD(SDLastFileResult()),0);
  END(*CASE*);
  WriteLn;
END ReportFIOResult;


(* Create a File of the size 100 chars and the name fn, e.g. f100.txt *)
PROCEDURE CreateFile100(fn: ARRAY OF CHAR);
  VAR written, i: INTEGER; s: ARRAY [0..255] OF CHAR; f: SDFile; readOnly, new, binary: BOOLEAN;
BEGIN
  readOnly := FALSE;
  new := TRUE;
  binary := FALSE;
  s := "123456789";
  SDLookup(f,fn,readOnly, new, binary); ReportFIOResult("SDLookup",fn);
  FOR i := 1 TO 10 DO
    written := SDFileWrite(f,ADR(s),Length(s)); written := SDFileWrite(f,ADR(SDEol),1);
  END(*FOR*);
  SDFlush(f); SDClose(f);
END CreateFile100;



(*---------------------------------*)
(*=====   Buffering of data   =====*)
(*---------------------------------*)

CONST
  bufferSize = 1024; (* as in DMFiles.mod of BatchDM *)
TYPE
  Buffer = ARRAY [1..bufferSize] OF CHAR; (* as in DMFiles.mod of BatchDM *)
VAR
  uses2ChEOL: BOOLEAN;

PROCEDURE WriteBuf (VAR(*speed-up*) b: ARRAY OF CHAR);
  CONST CR = 15C; LF = 12C;
  VAR i,n: INTEGER;
BEGIN (* WriteBuf *)
  i := 0; n := HIGH(b);
  WHILE (i<=n) AND (b[i]<>0C) DO
    IF uses2ChEOL AND (b[i]=SDEol) AND (i<n) AND (b[i+1]=SDEol2) THEN
      WriteString("EOL2"); WriteLn; INC(i);
    ELSIF (b[i]=CR) AND (i<n) AND (b[i+1]=LF) THEN
      WriteString("CRLF"); WriteLn; INC(i);
    ELSIF (b[i]=SDEol) THEN
      WriteString("EOL"); WriteLn;
    ELSIF (b[i]=CR) THEN
      WriteString("CR"); WriteLn;
```

```
    ELSIF (b[i]=LF) THEN
      WriteString("LF"); WriteLn;
    ELSE
      WriteChar(b[i]);
    END(*IF*);
    INC(i);
  END(*WHILE*);
END WriteBuf;

PROCEDURE CloseBuf (VAR b: ARRAY OF CHAR; i: INTEGER);
  VAR n: INTEGER;
BEGIN (* CloseBuf *)
  n := HIGH(b); (*. WriteString("HIGH(b) = "); WriteInt(n,0); WriteLn; .*)
  IF i<=n THEN b[i] := 0C END;
END CloseBuf;



(*****************************)
(*#####   Core Routines   #####*)
(*****************************)

PROCEDURE DocuTestSysDep;
BEGIN (* DocuTestSysDep *)
  WriteLn;
  WriteLn;
  WriteLn;
  WriteString("++++++++++++"); WriteLn;
  WriteString("TestSysDep:"); WriteLn;
  WriteString("++++++++++++"); WriteLn;
END DocuTestSysDep;



(************************)
(*#####   File IO   #####*)
(************************)

(*
    SDNullDevice      = -1;
    SDStdInput        = 0;
    SDStdOutput       = 1;
    SDStdError        = 2;
  VAR
    SDEol              : CHAR;                  (* readOnly variable *)
    SDEol2             : CHAR;                  (* readOnly variable *)
    SDDirSeperator     : CHAR;                  (* readOnly variable *)
    SDM2PathEnvVarName : ARRAY[0..10] OF CHAR;  (* readOnly variable *)


  TYPE
    SDFile          = INTEGER;
    SDFileResult    = (SDDone, SDFileNotFound, SDTooManyFiles,
                       SDNoWriteAccess, SDNoReadAccess,
                       SDFileNameWrong,
                       SDOtherResult);

  PROCEDURE SDLookup  (VAR file : SDFile; fileName: ARRAY OF CHAR;
                            readOnly, new, binary : BOOLEAN);
  PROCEDURE SDDelete  (fileName         : ARRAY OF CHAR);
  PROCEDURE SDRename  (oldName, newName : ARRAY OF CHAR);
  PROCEDURE SDClose   (VAR file : SDFile);
  PROCEDURE SDFlush   (    file : SDFile);
  PROCEDURE SDEOF     (    file : SDFile) : BOOLEAN;
  PROCEDURE SDTruncateFile(file : SDFile);
  PROCEDURE SDFileRead    (file : SDFile; buffer: ADDRESS; size: LONGCARD): LONGCARD;
  PROCEDURE SDFileWrite   (file : SDFile; buffer: ADDRESS; size: LONGCARD): LONGCARD;
  PROCEDURE SDGetFileSize (file : SDFile) : LONGCARD;
  PROCEDURE SDGetFileDates(file : SDFile; VAR create,modif: LONGINT);
  PROCEDURE SDGetFilePos  (file : SDFile) : LONGCARD;
  PROCEDURE SDSetFilePos  (file : SDFile; pos : LONGCARD);
```

```
  PROCEDURE SDLastFileResult () : SDFileResult;
*)


PROCEDURE OpenAnExistingMacFile(VAR testNo: INTEGER); (* Precondition: file TestMac.IN exists *)
  VAR f: SDFile; fn: ARRAY [0..255] OF CHAR;
  CONST readOnly = FALSE; new = FALSE; binary = FALSE;
    maxReadAttempts = 10;
  VAR size, maxSize: LONGCARD; b: Buffer; i: INTEGER;
  PROCEDURE CleanUp;
  BEGIN (* CleanUp *)
    SDClose(f); ReportFIOResult("SDClose",fn);
    WriteLn;
  END CleanUp;
BEGIN (* OpenAnExistingMacFile *)
  INC(testNo);
  WriteLn;
  WriteString("Test "); WriteInt(testNo,1);
  WriteString(": OpenAnExistingMacFile (Precondition: file TestMac.IN exists)"); WriteLn;
  WriteString("======================================================================"); WriteLn;
  fn := "TestMac.IN";
  SDLookup(f,fn,readOnly, new, binary); ReportFIOResult("SDLookup",fn);
  IF SDLastFileResult()<>SDDone THEN WriteLn; RETURN END;
  IF NOT SDEOF(f) THEN
    maxSize := bufferSize;
    WriteString("Read from file:"); WriteLn;
    WriteString("---------------"); WriteLn;
    i := 0;
    WHILE NOT SDEOF(f) AND (i<=maxReadAttempts) DO
      size := SDFileRead(f,ADR(b),maxSize); CloseBuf(b,size);
      WriteBuf(b);
      WriteString(" ("); WriteInt(size,1); WriteString(" chars read)");
      WriteLn;
      INC(i);
    END(*WHILE*);
    WriteString("---------------"); WriteLn;
  ELSE
    WriteString("Nothing read from file, since SDEOF returned TRUE"); WriteLn;
  END(*IF*);
  WriteString("File size returned by SDGetFileSize = "); WriteCard(SDGetFileSize(f),1); WriteLn;
  CleanUp;
  (* IF VERSION_ISO *)
  EXCEPT
    WriteLn; WriteString("#### EXCEPTION in OpenAnExistingMacFile encountered"); WriteLn;
    CleanUp;
  (* ENDIF VERSION_ISO *)
END OpenAnExistingMacFile;


PROCEDURE OpenAnExistingUnixFile(VAR testNo: INTEGER); (* Precondition: file TestUnix.IN exists *)
  VAR f: SDFile; fn: ARRAY [0..255] OF CHAR;
  CONST readOnly = FALSE; new = FALSE; binary = FALSE;
    maxReadAttempts = 10;
  VAR size, maxSize: LONGCARD; b: Buffer; i: INTEGER;
  PROCEDURE CleanUp;
  BEGIN (* CleanUp *)
    SDClose(f); ReportFIOResult("SDClose",fn);
    WriteLn;
  END CleanUp;
BEGIN (* OpenAnExistingUnixFile *)
  INC(testNo);
  WriteLn;
  WriteString("Test "); WriteInt(testNo,1);
  WriteString(": OpenAnExistingUnixFile (Precondition: file TestUnix.IN exists)"); WriteLn;
  WriteString("======================================================================"); WriteLn;
  fn := "TestUnix.IN";
  SDLookup(f,fn,readOnly, new, binary); ReportFIOResult("SDLookup",fn);
  IF SDLastFileResult()<>SDDone THEN WriteLn; RETURN END;
  IF NOT SDEOF(f) THEN
    maxSize := bufferSize;
```

```
    WriteString("Read from file:"); WriteLn;
    WriteString("---------------"); WriteLn;
    i := 0;
    WHILE NOT SDEOF(f) AND (i<=maxReadAttempts) DO
      size := SDFileRead(f,ADR(b),maxSize); CloseBuf(b,size);
      WriteBuf(b);
      WriteString(" ("); WriteInt(size,1); WriteString(" chars read)");
      WriteLn;
      INC(i);
    END(*WHILE*);
    WriteString("---------------"); WriteLn;
  ELSE
    WriteString("Nothing read from file, since SDEOF returned TRUE"); WriteLn;
  END(*IF*);
  WriteString("File size returned by SDGetFileSize = "); WriteCard(SDGetFileSize(f),1); WriteLn;
  CleanUp;
  (* IF VERSION_ISO *)
  EXCEPT
    WriteLn; WriteString("#### EXCEPTION in OpenAnExistingUnixFile encountered"); WriteLn;
    CleanUp;
  (* ENDIF VERSION_ISO *)
END OpenAnExistingUnixFile;


PROCEDURE OpenAnExistingDOSFile(VAR testNo: INTEGER); (* Precondition: file TestDOS.IN exists *)
  VAR f: SDFile; fn: ARRAY [0..255] OF CHAR;
  CONST readOnly = FALSE; new = FALSE; binary = FALSE;
    maxReadAttempts = 10;
  VAR size, maxSize: LONGCARD; b: Buffer; i: INTEGER;
  PROCEDURE CleanUp;
  BEGIN (* CleanUp *)
    SDClose(f); ReportFIOResult("SDClose",fn);
    WriteLn;
  END CleanUp;
BEGIN (* OpenAnExistingDOSFile *)
  INC(testNo);
  WriteLn;
  WriteString("Test "); WriteInt(testNo,1);
  WriteString(": OpenAnExistingDOSFile (Precondition: file TestDOS.IN exists)"); WriteLn;
  WriteString("======================================================================="); WriteLn;
  WriteString("not yet implemented"); WriteLn;
  (*. WriteLn; RETURN; .*)
  fn := "TestDOS.IN";
  SDLookup(f,fn,readOnly, new, binary); ReportFIOResult("SDLookup",fn);
  IF SDLastFileResult()<>SDDone THEN WriteLn; RETURN END;
  IF NOT SDEOF(f) THEN
    maxSize := bufferSize;
    WriteString("Read from file:"); WriteLn;
    WriteString("---------------"); WriteLn;
    i := 0;
    WHILE NOT SDEOF(f) AND (i<=maxReadAttempts) DO
      size := SDFileRead(f,ADR(b),maxSize); CloseBuf(b,size);
      WriteBuf(b);
      WriteString(" ("); WriteInt(size,1); WriteString(" chars read)");
      WriteLn;
      INC(i);
    END(*WHILE*);
    WriteString("---------------"); WriteLn;
  ELSE
    WriteString("Nothing read from file, since SDEOF returned TRUE"); WriteLn;
  END(*IF*);
  WriteString("File size returned by SDGetFileSize = "); WriteCard(SDGetFileSize(f),1); WriteLn;
  CleanUp;
  (* IF VERSION_ISO *)
  EXCEPT
    WriteLn; WriteString("#### EXCEPTION in OpenAnExistingDOSFile encountered"); WriteLn;
    CleanUp;
  (* ENDIF VERSION_ISO *)
END OpenAnExistingDOSFile;
```

```
PROCEDURE WriteToANewFile(VAR testNo: INTEGER); (* always overwrites *)
  VAR f: SDFile; fn: ARRAY [0..255] OF CHAR;
  CONST readOnly = FALSE; new = TRUE; binary = FALSE;
  VAR written: LONGCARD; s: ARRAY [0..255] OF CHAR;
  PROCEDURE CleanUp;
  BEGIN (* CleanUp *)
    SDClose(f); ReportFIOResult("SDClose",fn);
    WriteLn;
  END CleanUp;
BEGIN (* WriteToANewFile *)
  INC(testNo);
  WriteLn;
  WriteString("Test "); WriteInt(testNo,1); WriteString(": WriteToANewFile (always overwrites)"); WriteLn;
  WriteString("=========================================="); WriteLn;
  fn := "Test.OUT";
  SDLookup(f,fn,readOnly, new, binary); ReportFIOResult("SDLookup",fn);
  IF SDLastFileResult() = SDDone THEN
    (* line 1 *)
    s := "Some text written to a file in its 1st line.";
    WriteString("Writing line 1 '"); WriteString(s); WriteString("'"); WriteLn;
    written := SDFileWrite(f,ADR(s),Length(s)); written := SDFileWrite(f,ADR(SDEol),1);
    WriteString("Written line 1 with "); WriteInt(Length(s)+1,1);
    WriteString(" bytes (with EOL at end)"); WriteLn;
    (* line 2 *)
    s := "Some more text written to a file in 2nd row.";
    WriteString("Writing line 2 '"); WriteString(s); WriteString("'"); WriteLn;
    written := SDFileWrite(f,ADR(s),Length(s)); written := SDFileWrite(f,ADR(SDEol),1);
    WriteString("Written line 2 with "); WriteInt(Length(s)+1,1);
    WriteString(" bytes (with EOL at end)"); WriteLn;
  ELSE
    WriteString("Nothing written to file '");
    WriteString(fn); WriteString("'");
    WriteString(", since creation failed"); WriteLn;
  END(*IF*);
  WriteString("File size returned by SDGetFileSize before flush = "); WriteCard(SDGetFileSize(f),1); WriteLn;
  SDFlush(f); ReportFIOResult("SDFlush",fn);
  WriteString("File size returned by SDGetFileSize after flush = "); WriteCard(SDGetFileSize(f),1); WriteLn;
  CleanUp;
  (* IF VERSION_ISO *)
  EXCEPT
    WriteLn; WriteString("#### EXCEPTION in WriteToANewFile encountered"); WriteLn;
    CleanUp;
  (* ENDIF VERSION_ISO *)
END WriteToANewFile;


PROCEDURE MakeRandomFileAccess(VAR testNo: INTEGER); (* Precondition: file Test.OUT exists *)
  VAR f: SDFile; fn: ARRAY [0..255] OF CHAR;
  CONST readOnly = TRUE; new = FALSE; binary = FALSE;
  VAR size, maxSize: LONGCARD; b: Buffer; i: INTEGER;
  PROCEDURE JumpToAndRead (pos: LONGCARD);
  BEGIN (* JumpToAndRead *)
    WriteString("---------------"); WriteLn;
    SDSetFilePos(f,pos); ReportFIOResult("SDSetFilePos",fn);
    IF SDGetFilePos(f)=pos THEN
      (* Attempt to read only if positioning successful *)
      WriteString("Read from file after pos "); WriteCard(pos,2); WriteString(": '");
        size := SDFileRead(f,ADR(b),maxSize); CloseBuf(b,size);
        WriteBuf(b);
        WriteString("' ("); WriteInt(size,1); WriteString(" chars read)");
    ELSE
      WriteString("#### Failed attempt to set the file position to position ");
      WriteCard(pos,1); WriteLn;
      WriteString("     (Note: Above error condition is encountered on purpose"); WriteLn;
      WriteString("     and does not represent an error in SysDep. On the contrary,"); WriteLn;
      WriteString("     it would be an error of SysDep if this message would not be"); WriteLn;
      WriteString("     displayed.)");
    END(*IF*);
    WriteLn;
  END JumpToAndRead;
  PROCEDURE CleanUp;
```

```
  BEGIN (* CleanUp *)
    SDClose(f); ReportFIOResult("SDClose",fn);
    WriteLn;
  END CleanUp;
BEGIN (* MakeRandomFileAccess *)
  INC(testNo);
  WriteLn;
  WriteString("Test "); WriteInt(testNo,1);
  WriteString(": MakeRandomFileAccess (Precondition: file Test.OUT exists)"); WriteLn;
  WriteString("======================================================================"); WriteLn;
  fn := "Test.OUT";
  SDLookup(f,fn,readOnly, new, binary); ReportFIOResult("SDLookup",fn);
  IF SDLastFileResult()<>SDDone THEN WriteLn; RETURN END;
  IF NOT SDEOF(f) THEN
    maxSize := bufferSize;
    JumpToAndRead(   5);
    JumpToAndRead(  80);
    JumpToAndRead(   2);
    JumpToAndRead(2000);
    (* With P1 there is now the file in an inconsistent state, i.e. RawRead is no longer available.
    Thus any attempt to read again after a failed positioning command leads to an exception. *)
    (*. JumpToAndRead(   1); .*)
  ELSE
    WriteString("Nothing read from file, since SDEOF returned TRUE"); WriteLn;
  END(*IF*);
  CleanUp;
  (* IF VERSION_ISO *)
  EXCEPT
    WriteLn; WriteString("#### EXCEPTION in MakeRandomFileAccess encountered"); WriteLn;
    CleanUp;
  (* ENDIF VERSION_ISO *)
END MakeRandomFileAccess;


PROCEDURE AlterIOMode(VAR testNo: INTEGER);
  VAR s, fn : ARRAY [0..255] OF CHAR; f : SDFile;
  CONST readOnly = FALSE; new = TRUE; binary = FALSE;
    maxReadAttempts = 10;
  VAR maxSize: LONGCARD; b: Buffer; size, written, i,j: INTEGER; k: CARDINAL;
  PROCEDURE CleanUp;
  BEGIN (* CleanUp *)
    SDClose(f); ReportFIOResult("SDClose",fn);
    WriteLn;
  END CleanUp;
BEGIN
  INC(testNo);
  WriteLn;
  WriteString("Test "); WriteInt(testNo,1);
  WriteString(": AlterIOMode, usage of read an write access simultaneously"); WriteLn;
  WriteString("======================================================================"); WriteLn;
  fn := "AlterIO.OUT";
  SDLookup(f,fn,readOnly, new, binary); ReportFIOResult("SDLookup",fn);
  IF SDLastFileResult() = SDDone THEN
    (* write into the file *)
    (* line 1 *)
    s := "Some text written to a file in its 1st line.";
    size := Length(s) + 1;
    WriteString("Writing line 1 '"); WriteString(s); WriteString("'"); WriteLn;
    written := SDFileWrite(f,ADR(s),Length(s)); written := SDFileWrite(f,ADR(SDEol),1);
    WriteString("Written line 1 with "); WriteInt(Length(s)+1,1);
    WriteString(" bytes (with EOL at end)"); WriteLn;
    (* line 2 *)
    s := "Some more text written to a file in 2nd row.";
    size := size + Length(s) + 1;
    WriteString("Writing line 2 '"); WriteString(s); WriteString("'"); WriteLn;
    written := SDFileWrite(f,ADR(s),Length(s)); written := SDFileWrite(f,ADR(SDEol),1);
    WriteString("Written line 2 with "); WriteInt(Length(s)+1,1);
    WriteString(" bytes (with EOL at end)"); WriteLn;
    WriteString("File size returned by SDGetFileSize before flush = "); WriteCard(SDGetFileSize(f),1); WriteLn;
    SDFlush(f); ReportFIOResult("SDFlush",fn);
    WriteString("File size returned by SDGetFileSize after flush = "); WriteCard(SDGetFileSize(f),1); WriteLn;
```

```
  (* read from the file *)
  (* should be EOF *)
  IF NOT SDEOF(f) THEN
     WriteString("File '"); WriteString(fn); WriteString("' should be EOF after writing"); WriteLn;
  END(*IF*);
  WriteString("Repositioning file to its begin to read what was written without close"); WriteLn;
  SDSetFilePos(f, 0);
  maxSize := HIGH(s)-1;
  WriteString("Read from file:"); WriteLn;
  WriteString("---------------"); WriteLn;
  i := 0;
  WHILE NOT SDEOF(f) AND (i<=maxReadAttempts) DO
    size := SDFileRead(f,ADR(s),maxSize); CloseBuf(s,size);
    WriteBuf(s);
    WriteString(" ("); WriteInt(size,1); WriteString(" chars read)");
    WriteLn;
    INC(i);
  END(*WHILE*);
  WriteString("---------------"); WriteLn;
  IF NOT SDEOF(f) THEN
    WriteString("File '"); WriteString(fn); WriteString("' should be EOF after reading"); WriteLn;
  ELSE
    WriteString("Now at EOF and resuming writing:"); WriteLn;
  END(*IF*);
  (* line 3 *)
  s := "Some more text written to a file in 3rd row.";
  size := size + Length(s) + 1;
  WriteString("Writing line 3 '"); WriteString(s); WriteString("'"); WriteLn;
  written := SDFileWrite(f,ADR(s),Length(s)); written := SDFileWrite(f,ADR(SDEol),1);
  WriteString("Written line 3 with "); WriteInt(Length(s)+1,1);
  WriteString(" bytes (with EOL at end)"); WriteLn;
  FOR k:= 4 TO 24 DO
    s := "Some more text written to a file in "; j := 0;
    CopyString(s,0,stopAt0C,b,j);
    s[0] := CHR(ORD("0") + k DIV 10); s[1] := CHR(ORD("0") + k MOD 10); s[2] := 0C;
    CopyString(s,0,stopAt0C,b,j);
    s := ". row.";
    CopyString(s,0,stopAt0C,b,j);
    written := SDFileWrite(f,ADR(b),Length(b)); written := SDFileWrite(f,ADR(SDEol),1);
    WriteString("Writing '"); WriteString(b); WriteString("'"); WriteLn;
  END(*FOR*);
  SDFlush(f); ReportFIOResult("SDFlush",fn);
  (* read from the file *)
  (* should be EOF *)
  IF NOT SDEOF(f) THEN
     WriteString("File '"); WriteString(fn); WriteString("' should be EOF after writing"); WriteLn;
  END(*IF*);
  SDSetFilePos(f, 0);
  maxSize := bufferSize;
  WriteString("Read from file:"); WriteLn;
  WriteString("---------------"); WriteLn;
  i := 0; WriteString("'");
  WHILE NOT SDEOF(f) AND (i<=maxReadAttempts) DO
    size := SDFileRead(f,ADR(b),maxSize); CloseBuf(b,size);
    WriteBuf(b);
    WriteString("' ("); WriteInt(size,1); WriteString(" chars read)");
    WriteLn;
    IF NOT SDEOF(f) THEN WriteString("'") END;
    INC(i);
  END(*WHILE*);
  WriteString("---------------"); WriteLn;
ELSE
  WriteString("Nothing written to file '");
  WriteString(fn); WriteString("'");
  WriteString(", since creation failed"); WriteLn;
END(*IF*);
CleanUp;
(* IF VERSION_ISO *)
EXCEPT
  WriteLn; WriteString("#### EXCEPTION in AlterIOMode encountered"); WriteLn;
  CleanUp;
```

```
  (* ENDIF VERSION_ISO *)
END AlterIOMode;


PROCEDURE TruncateFile (VAR testNo: INTEGER);
  VAR readOnly, new, binary: BOOLEAN; f: SDFile; fn: ARRAY [0..255] OF CHAR;
  PROCEDURE CleanUp;
  BEGIN (* CleanUp *)
    SDClose(f); ReportFIOResult("SDClose",fn);
    WriteLn;
  END CleanUp;
BEGIN (* TruncateFile *)
  INC(testNo);
  WriteLn;
  WriteString("Test "); WriteInt(testNo,1); WriteString(": Truncate file (uses f100.txt)"); WriteLn;
  WriteString("====================================="); WriteLn;
  readOnly := TRUE;
  new := FALSE;
  binary := FALSE;
  fn := "f100_trunc.txt";
  CreateFile100(fn);
  SDLookup(f, fn , readOnly, new, binary); ReportFIOResult("SDLookup",fn);
  SDSetFilePos(f, 50);
  IF SDGetFilePos(f) # 50 THEN
    WriteString("#### Unexpected error: We should be at position 50, but we are at: ");
    WriteCard(SDGetFilePos(f), 1); WriteLn();
  END(*IF*);
  SDTruncateFile(f); ReportFIOResult("SDTruncateFile", fn);
  IF SDGetFileSize(f) # 50 THEN
    WriteString("#### Unexpected error: File should be truncated at position 50, size is: ");
    WriteCard(SDGetFileSize(f), 1); WriteLn();
  END(*IF*);
  IF SDGetFilePos(f) # 50 THEN
    WriteString("#### Unexpected error: We should be at position 50, but we are at: ");
    WriteCard(SDGetFilePos(f), 1); WriteLn();
  END(*IF*);
  SDSetFilePos(f, 0);
  IF SDGetFilePos(f) # 0 THEN
    WriteString("#### Unexpected error: We should be at position 0, but we are at: ");
    WriteCard(SDGetFilePos(f), 1); WriteLn();
  END(*IF*);
  SDTruncateFile(f); ReportFIOResult("SDTruncateFile", fn);
  IF SDGetFileSize(f) # 0 THEN
    WriteString("#### Unexpected error: File should be truncated at position 0, size is: ");
    WriteCard(SDGetFileSize(f), 1); WriteLn();
  END(*IF*);
  IF SDGetFilePos(f) # 0 THEN
    WriteString("#### Unexpected error: We should be at position 0, but we are at: ");
    WriteCard(SDGetFilePos(f), 1); WriteLn();
  END(*IF*);
  CleanUp;
  (* IF VERSION_ISO *)
  EXCEPT
    WriteLn; WriteString("#### EXCEPTION in TruncateFile encountered"); WriteLn;
    CleanUp;
  (* ENDIF VERSION_ISO *)
END TruncateFile;



PROCEDURE RenameDeleteFiles (VAR testNo: INTEGER);
  CONST afilePos = 22;
  VAR readOnly, new, binary: BOOLEAN; f: SDFile; fn, fn2: ARRAY [0..255] OF CHAR;
  PROCEDURE CleanUp;
  BEGIN (* CleanUp *)
    IF f<>SDNullDevice THEN SDClose(f) END; ReportFIOResult("SDClose",fn);
    WriteLn;
  END CleanUp;
BEGIN (* RenameDeleteFiles *)
  INC(testNo);
  WriteLn;
```

```
  WriteString("Test "); WriteInt(testNo,1); WriteString(": Renaming / Deleting of files"); WriteLn;
  WriteString("===================================="); WriteLn;
  readOnly := TRUE;
  new := FALSE;
  binary := FALSE;
  fn := "f100_o.txt";
  fn2 := "f100_n.txt";
  WriteString("Creating a test file '"); WriteString(fn); WriteString("':"); WriteLn;
  CreateFile100(fn);
  WriteString("Test presence of created file '"); WriteString(fn); WriteString("':"); WriteLn;
  SDLookup(f, fn , readOnly, new, binary); ReportFIOResult("SDLookup",fn);
  WriteLn;

  WriteString("Setting file position to "); WriteCard(afilePos,1);
  WriteString(" (in open file before renaming)"); WriteLn;
  SDSetFilePos(f,afilePos);
  IF SDGetFilePos(f) <> afilePos THEN
    WriteString("#### Unexpected error: File position should be "); WriteCard(afilePos,1);
    WriteString(" but is: ");
    WriteCard(SDGetFilePos(f), 1); WriteLn();
  END(*IF*);
  WriteString("Renaming file '"); WriteString(fn); WriteString("' (open) to '");
  WriteString(fn2); WriteString("':"); WriteLn;
  SDRename(fn, fn2); ReportFIOResult("SDRename",fn);
  IF SDGetFilePos(f) = afilePos THEN
    WriteString("File position "); WriteCard(afilePos,1);
    WriteString(" in open file was preserved as expected"); WriteLn;
  ELSE
    WriteString("#### Unexpected error: File position after renaming should still be ");
    WriteCard(afilePos,1); WriteString(" but is: ");
    WriteCard(SDGetFilePos(f), 1); WriteLn();
  END(*IF*);
  SDClose(f); ReportFIOResult("SDClose",fn2);
  fn := "f100.txt";
  WriteString("Test presence of file renamed to '"); WriteString(fn2); WriteString("':"); WriteLn;
  SDLookup(f, fn2 , readOnly, new, binary); ReportFIOResult("SDLookup (readOnly)",fn2);
  SDClose(f); ReportFIOResult("SDClose",fn2);

  WriteLn;
  WriteString("Renaming file '"); WriteString(fn2); WriteString("' (closed) to '");
  WriteString(fn); WriteString("':"); WriteLn;
  SDRename(fn2, fn); ReportFIOResult("SDRename",fn2);
  WriteString("Test presence of file renamed to '"); WriteString(fn); WriteString("':"); WriteLn;
  SDLookup(f, fn , readOnly, new, binary); ReportFIOResult("SDLookup (readOnly)",fn);
  SDClose(f); ReportFIOResult("SDClose",fn);
  WriteLn;
  WriteString("Deleting file '"); WriteString(fn); WriteString("':"); WriteLn;
  SDDelete(fn); ReportFIOResult("SDDelete",fn);
  WriteString("Test wether file '"); WriteString(fn); WriteString("' was actually deleted:"); WriteLn;
  SDLookup(f, fn , readOnly, new, binary); ReportFIOResult("SDLookup",fn);
  CleanUp;
  (* IF VERSION_ISO *)
  EXCEPT
    WriteLn; WriteString("#### EXCEPTION in RenameDeleteFiles encountered"); WriteLn;
    CleanUp;
  (* ENDIF VERSION_ISO *)
END RenameDeleteFiles;



PROCEDURE WriteStdChans(VAR testNo: INTEGER);
VAR str, str2 : ARRAY [0..127] OF CHAR; size, written: INTEGER;
BEGIN
  INC(testNo);
  WriteLn;
  WriteString("Test "); WriteInt(testNo,1);
  WriteString(": Write to StdOutput and StdError and read StdIn, opened by SysDep"); WriteLn;
  WriteString("======================================================================="); WriteLn;
  str := "This text should be on StdOut";
  written := SDFileWrite(SDStdOutput, ADR(str), Length(str)); written := SDFileWrite(SDStdOutput,ADR(SDEol),1);
  SDFlush(SDStdOutput);
```

```
  str := "This text should be on StdErr";
  written := SDFileWrite(SDStdError, ADR(str), Length(str)); written := SDFileWrite(SDStdError,ADR(SDEol),1);
  SDFlush(SDStdError);
  str:= "Please write some text (Ctrl-d if finished): ";
  written := SDFileWrite(SDStdOutput, ADR(str), Length(str)); SDFlush(SDStdOutput);
  size := SIZE(str2);
  size := SDFileRead(SDStdInput, ADR(str2), size); CloseBuf(str2,size);
  str := "You wrote: ";
  written := SDFileWrite(SDStdOutput,ADR(SDEol),1); written := SDFileWrite(SDStdOutput, ADR(str), Length(str));
  written := SDFileWrite(SDStdOutput, ADR(str2), Length(str2)); written := SDFileWrite(SDStdOutput,ADR(SDEol),1);
  SDFlush(SDStdOutput);
END WriteStdChans;



(****************************************)
(*#####   Dynamic memory i.e. Heap   #####*)
(****************************************)

(*
  PROCEDURE SDAlloc   (VAR a : ADDRESS; byteCount: LONGCARD);
  PROCEDURE SDDealloc (VAR a : ADDRESS);
  PROCEDURE SDValidPtr(a : ADDRESS) : BOOLEAN;
*)

PROCEDURE HeapManagement (VAR testNo: INTEGER);
  VAR a, b : ADDRESS;
BEGIN (* HeapManagement *)
  INC(testNo);
  WriteLn;
  WriteString("Test "); WriteInt(testNo,1); WriteString(": Heap Management"); WriteLn;
  WriteString("======================="); WriteLn;
  (* IF VERSION_EPC *) (*.
  a := NIL; WriteString("a := NIL, since epc Modula-2 crashes with uninitialized pointer!"); WriteLn;
  .*) (* ENDIF VERSION_EPC *)
  WriteString("Calling SDValidPtr(a) with never inited ADDRESS 'a' (local variable):"); WriteLn;
  IF SDValidPtr(a) THEN
    WriteString("SDValidPtr: Not initialised ADDRESS 'a' is a valid pointer"); WriteLn;
  ELSE
    WriteString("SDValidPtr: Not initialised ADDRESS 'a' is not a valid pointer"); WriteLn;
  END(*IF*);
  SDAlloc(a, 255);
  IF SDValidPtr(a) THEN
    WriteString("SDAlloc: Allocated ADDRESS 'a' is a valid pointer"); WriteLn;
  ELSE
    WriteString("SDAlloc: Allocated ADDRESS 'a' is not a valid pointer"); WriteLn;
  END(*IF*);
  b := a;
  SDDealloc(a);
  IF SDValidPtr(a) THEN
    WriteString("SDDealloc: Deallocated ADDRESS 'a' is a valid pointer"); WriteLn;
  ELSE
    WriteString("SDDealloc: Deallocated ADDRESS 'a' is not a valid pointer"); WriteLn;
  END(*IF*);
  IF a=NIL THEN
    WriteString("SDDealloc: As expected ADDRESS 'a' is NIL");
  ELSE
    WriteString("Unexpected result from SDDealloc(a): ADDRESS 'a' is different from NIL"); WriteLn;
    IF a=b THEN
      WriteString("ADDRESS 'a' has still the same value before the call to SDDealloc(a)");
    ELSE
      WriteString("ADDRESS 'a' has another value than that before the call to SDDealloc(a)");
    END(*IF*);
  END(*IF*);
  WriteLn;
  SDDealloc(a);
  IF a = NIL THEN
    IF SDValidPtr(a) THEN
      WriteString("SDValidPtr: ADDRESS 'a' = NIL erroneously considered a valid pointer"); WriteLn;
    ELSE
      WriteString("SDValidPtr: ADDRESS 'a' = NIL correctly considered as an invalid pointer"); WriteLn;
```

```
    END(*IF*);
  END(*IF*);
  WriteLn;
  (* IF VERSION_ISO *)
  EXCEPT
    WriteLn; WriteString("#### EXCEPTION in HeapManagement encountered"); WriteLn;
  (* ENDIF VERSION_ISO *)
END HeapManagement;


(*************************************)
(******   Clock and File Dates   ******)
(*************************************)

(*
  CONST
    SDJan = 1; SDFeb = 2; SDMar = 3; SDApr = 4;  SDMai = 5;  SDJun = 6;
    SDJul = 7; SDAug = 8; SDSep = 9; SDOct = 10; SDNov = 11; SDDec = 12;
    SDSun = 1; SDMon = 2; SDTue = 3; SDWed = 4;  SDThu = 5;  SDFri = 6; SDSat = 7;
  (* hour is always within range [0..23] *)

  TYPE
    SDTime; (* UTC - Universal Time Coordinated = GMT - Greenwich Mean Time
               see e.g. http://en.wikipedia.org/wiki/UTC *)
    SDDate = RECORD
      year,   (* year < 50 -> + 2000 ; > 50 -> + 1900 *)
      month,  (* SDJan .. SDDec *)
      day,    (* 1 .. 31 day of month *)
      hour,   (* 0 .. 23 *)
      minute, (* 0 .. 59 *)
      second, (* 0 .. 59 *)
      sec100, (* 0 .. 99 *)
      dayOfWeek   : INTEGER; (* SDSun .. SDSat *)
      daylightSaving: BOOLEAN;
      timeZoneDiff: INTEGER; (* in minutes relative to UTC, e.g. +60 for Central European Time *)
    END;(*RECORD*)

  PROCEDURE SDGetTime    (VAR time : SDTime);
  PROCEDURE SDTimeToDate (    time : SDTime; VAR date : SDDate);
  PROCEDURE SDDateToTime (    date : SDDate; VAR time : SDTime);
  (* SDDateToTime requires timeZoneDiff to be set properly, e.g.
  by first calling SDGetTime and then SDTimeToDate *)
*)

PROCEDURE GetFileAttributesAndClock(VAR testNo: INTEGER); (* Precondition: files TestMac.IN and Test.OUT exist *)
  VAR f: SDFile; fn: ARRAY [0..255] OF CHAR;
  CONST readOnly = TRUE; new = FALSE; binary = FALSE; fw = 12;
  VAR create,modif: SDTime; createI,modifI: LONGINT; time: SDTime; sdDate: SDDate; tzDiff: REAL;
  PROCEDURE WriteDateTime (dt: SDTime);
    VAR d: SDDate; t: SDTime;
  BEGIN (* WriteDateTime *)
    t := SDTime(dt); (* Opaque SysDep.SDTime is implemented as LONGCARD *)
    SDTimeToDate(t,d);
    WriteString("  "); IF d.day<10    THEN WriteChar("0") END; WriteCard(d.day,1);
                       WriteString("/");
                       IF d.month<10  THEN WriteChar("0") END; WriteCard(d.month,1);
                       WriteString("/");
                       WriteCard(d.year,4);
    WriteString("  "); IF d.hour<10   THEN WriteChar("0") END; WriteCard(d.hour,1);
                       WriteString("h");
                       IF d.minute<10 THEN WriteChar("0") END; WriteCard(d.minute,1);
                       WriteString(":");
                       IF d.second<10 THEN WriteChar("0") END; WriteCard(d.second,1);
  END WriteDateTime;
  PROCEDURE CleanUp;
  BEGIN (* CleanUp *)
    SDClose(f); ReportFIOResult("SDClose",fn);
    WriteLn;
  END CleanUp;
BEGIN (* GetFileAttributesAndClock *)
  INC(testNo);
```

```
  WriteLn;
  WriteString("Test "); WriteInt(testNo,1);
  WriteString("a: GetFileAttributesAndClock (files ZFile.MOD, TestMac.IN exist)"); WriteLn;
  WriteString("======================================================================="); WriteLn;
  fn := "ZFile.MOD";
  SDGetFileData(fn,create,modif); ReportFIOResult("SDGetFileData",fn);
  IF SDLastFileResult()=SDDone THEN
    WriteString("Creation date:             "); WriteCard(LONGCARD(create),fw); WriteDateTime(SDTime(create)); WriteLn;
    WriteString("Modification date:         "); WriteCard(LONGCARD(modif) ,fw); WriteDateTime(SDTime(modif )); WriteLn;
  ELSE
    WriteString("Could not get attributes from file '"); WriteString(fn); WriteString("'"); WriteLn;
  END(*IF*);
  WriteLn;

  fn := "TestMac.IN";
  SDLookup(f,fn,readOnly, new, binary); ReportFIOResult("SDLookup",fn);
  IF SDLastFileResult()<>SDDone THEN WriteLn; RETURN END;
  SDGetFileDates(f,createI,modifI); ReportFIOResult("SDGetFileDates",fn);
  create := SDTime(createI); modif := SDTime(modifI);
  IF SDLastFileResult()=SDDone THEN
    WriteString("Creation date:             "); WriteCard(LONGCARD(create),fw); WriteDateTime(SDTime(create)); WriteLn;
    WriteString("Modification date:         "); WriteCard(LONGCARD(modif ),fw); WriteDateTime(SDTime(modif )); WriteLn;
  ELSE
    WriteString("Could not get attributes from file '"); WriteString(fn); WriteString("'"); WriteLn;
  END(*IF*);
  CleanUp;

  WriteString("Test "); WriteInt(testNo,1); WriteString("b: GetFileAttributesAndClock Time routines"); WriteLn;
  WriteString("================================================="); WriteLn;
  SDGetTime(time);
  WriteString("SDGetTime returned:          "); WriteCard(LONGCARD(time),1);
  WriteDateTime(time); WriteLn;
  (* reconversion *)
  SDTimeToDate(time,sdDate);
  SDDateToTime(sdDate,time);
  WriteString("SDDateToTime<>SDDateToTime:  "); WriteCard(LONGCARD(time),1);
  WriteDateTime(time); WriteLn;
  (* reconvert once more *)
  SDTimeToDate(time,sdDate);
  CASE  sdDate.dayOfWeek OF
  | SDSun: WriteString("Sunday");
  | SDMon: WriteString("Monday");
  | SDTue: WriteString("Tuesday");
  | SDWed: WriteString("Wedensday");
  | SDThu: WriteString("Thursday");
  | SDFri: WriteString("Friday");
  | SDSat: WriteString("Saturday");
  ELSE
    WriteString("#### Error: Bad sdDate.dayOfWeek ORD(sdDate.dayOfWeek) = "); WriteInt(ORD(sdDate.dayOfWeek),1);
  END(*CASE*);
  WriteDateTime(time); WriteLn;
  WriteString("Daylight saving time: ");
  IF sdDate.daylightSaving THEN WriteString("ON") ELSE WriteString("OFF") END;
  WriteLn;
  tzDiff := FLOAT(sdDate.timeZoneDiff)/ 60.0;
  IF tzDiff<>0.0 THEN
    IF tzDiff>0.0 THEN WriteString("+") ELSE WriteString("-") END;
  END(*IF*);
  WriteFixed(tzDiff,1,3);
  WriteString(" hour(s) difference to UTC (Coordinated Universal Time)"); WriteLn;
  WriteString("100th seconds: "); WriteInt(sdDate.sec100,1); WriteLn;
  WriteLn;
  (* IF VERSION_ISO *)
  EXCEPT
    WriteLn; WriteString("#### EXCEPTION in GetFileAttributesAndClock encountered"); WriteLn;
    CleanUp;
  (* ENDIF VERSION_ISO *)
END GetFileAttributesAndClock;
```

```
(*****************************)
(******   Directories   *******)
(*****************************)

(*
  TYPE
    SDDirectoryProc = PROCEDURE (INTEGER, ARRAY OF CHAR, BOOLEAN, VAR BOOLEAN);
                              (* itemNo,  filename,      isDir,   exit loop *)

  PROCEDURE SDCurWorkDirectory(VAR path: ARRAY OF CHAR);
  PROCEDURE SDCreateDir( path, dirN: ARRAY OF CHAR;  VAR done: BOOLEAN );
  PROCEDURE SDDirInfo( path, dirN: ARRAY OF CHAR;
                       VAR dirExists, containsFiles : BOOLEAN );
  PROCEDURE SDDoForAllFilesInDirectory(path: ARRAY OF CHAR; dp: SDDirectoryProc);
  (* Note: Hidden files, e.g. those whose names start with ".", may be detected too! *)
*)

PROCEDURE DoForAllItemsInDir (no: INTEGER; fn: ARRAY OF CHAR; isDir: BOOLEAN; VAR exit: BOOLEAN);
BEGIN (* DoForAllItemsInDir *)
  WriteString("# "); WriteInt(no,2); WriteString(": '"); WriteString(fn); WriteString("' ");
  IF isDir THEN
    WriteString("directory");
  ELSE
    WriteString("file");
  END(*IF*);
  WriteLn;
  exit := FALSE;
END DoForAllItemsInDir;


PROCEDURE DirectoryFunctions (VAR testNo: INTEGER);
  VAR path,dn, pfn: ARRAY [0..255] OF CHAR; i: INTEGER;
    dummy: ARRAY [0..255] OF CHAR; done: BOOLEAN;
    dirExists,containsFiles: BOOLEAN;
BEGIN (* DirectoryFunctions *)
  INC(testNo);
  WriteLn;
  WriteString("Test "); WriteInt(testNo,1); WriteString(": Directory functions"); WriteLn;
  WriteString("============================"); WriteLn;

  (* SDCurWorkDirectory *)
  dummy := "current dir";
  SDCurWorkDirectory(path);
  WriteString("SDCurWorkDirectory: '"); WriteString(path); WriteString("'"); WriteLn;

  (* SDCreateDir *)
  dn := "ADir";
  SDCreateDir(path,dn,done);
  IF done THEN
    WriteString("SDCreateDr: returned done '"); WriteString(dn); WriteString("'"); WriteLn;
  ELSE
    WriteString("SDCreateDr: returned not done '"); WriteString(dn); WriteString("'"); WriteLn;
  END(*IF*);

  (* SDDirInfo *)
  SDDirInfo(path,dn,dirExists,containsFiles); ReportFIOResult("SDDirInfo",dn);
  IF dirExists THEN
    WriteString("SDDirInfo: detected directory '"); WriteString(dn); WriteString("'"); WriteLn;
  ELSE
    WriteString("SDDirInfo: directory seems to be missing '"); WriteString(dn); WriteString("'"); WriteLn;
  END(*IF*);
  IF containsFiles THEN
    WriteString("SDDirInfo: detected file(s) in directory '"); WriteString(dn); WriteString("'"); WriteLn;
  ELSE
    WriteString("SDDirInfo: no files in dir '"); WriteString(dn); WriteString("'"); WriteLn;
  END(*IF*);

  (* SDDoForAllFilesInDirectory *)
  CompletePathFileName(path,dn,pfn);
  WriteString("SDDoForAllFilesInDirectory: for dir '"); WriteString(pfn); WriteString("'"); WriteLn;
  SDDoForAllFilesInDirectory(pfn,DoForAllItemsInDir); ReportFIOResult("SDDoForAllFilesInDirectory",pfn);
```

```
  WriteLn;

  (* IF VERSION_ISO *)
  EXCEPT
    WriteLn; WriteString("#### EXCEPTION in DirectoryFunctions encountered"); WriteLn;
  (* ENDIF VERSION_ISO *)
END DirectoryFunctions;



(**********************************)
(*#####   Hardware / Screen   ######*)
(**********************************)

(*
  CONST
    SDMacStr     = "Macintosh";
    SDUnixStr    = "Unix";
    SDIBMPCStr   = "IBM-PC";

  PROCEDURE SDGetComputerName(VAR name: ARRAY OF CHAR);

  CONST
    SDMC680x0Str    = "MC680x0";
    SDPPCStr        = "PPC";
    SDIntelStr      = "Intel";
    SDSPARCStr      = "SPARC";

  PROCEDURE SDGetCPUName(VAR name: ARRAY OF CHAR);


  CONST  (* consistent with DMSystem *)
    SDMacII  =   6;
    SDSUN    = 101;
    SDIBMPC  = 201;

  PROCEDURE SDComputerSystem(): INTEGER;

  (*=====   Single main screen   =====*)
  PROCEDURE SDScreenWidth()   : INTEGER; (* of main screen *)
  PROCEDURE SDScreenHeight()  : INTEGER; (* of main screen including menu bar *)
  PROCEDURE SDMenuBarHeight() : INTEGER;
  (*=====   Multiple screens   =====*)
  PROCEDURE SDMainScreen(): INTEGER;  (* is screen which contains menu bar *)
  PROCEDURE SDHowManyScreens(): INTEGER;
  PROCEDURE SDSuperScreen(VAR whichScreen, x,y,w,h, nrOfColors: INTEGER;
                          colorPriority: BOOLEAN);
  PROCEDURE SDGetScreenSize(screen: INTEGER; VAR x,y,w,h: INTEGER);
  (*=====   GUI Objects Sizes   =====*)
  PROCEDURE SDTitleBarHeight(): INTEGER;
  PROCEDURE SDScrollBarWidth(): INTEGER;
  PROCEDURE SDGrowIconSize(): INTEGER;   (* icon at the lower right corner of a window *)
  (*=====   Colors   =====*)
  PROCEDURE SDNumberOfColors(): INTEGER;
  PROCEDURE SDNumberOfColorsOnScreen(screen: INTEGER): INTEGER;
*)


PROCEDURE HardwareAndScreen (VAR testNo: INTEGER);
BEGIN (* HardwareAndScreen *)
  INC(testNo);
  WriteLn;
  WriteString("Test "); WriteInt(testNo,1); WriteString(": Hardware and Screen"); WriteLn;
  WriteString("============================"); WriteLn;

   (*.
    SDExecute testing. Could for instance use standard input to ask user for a command
    and execute it. If Ctrl-d is entered, the loop of SDExecute stops.
  .*)

  WriteLn;
```

```
  (* IF VERSION_ISO *)
  EXCEPT
    WriteLn; WriteString("#### EXCEPTION in HardwareAndScreen encountered"); WriteLn;
  (* ENDIF VERSION_ISO *)
END HardwareAndScreen;



(*****************************)
(*#####   Environment   #####*)
(*****************************)

(*
  PROCEDURE SDGetArgCount  (): CARDINAL;
  PROCEDURE SDGetArgument  (argNum : CARDINAL; VAR argument: ARRAY OF CHAR);
  PROCEDURE SDGetEnvVar    (varName : ARRAY OF CHAR; VAR varValue : ARRAY OF CHAR);
  PROCEDURE SDGetEnvVarListElem(    varName : ARRAY OF CHAR;
                                    index   : INTEGER;
                                VAR elemValue : ARRAY OF CHAR);
*)

PROCEDURE AccessEnvironment (VAR testNo: INTEGER);
  PROCEDURE RetrieveEnvVar (varName: ARRAY OF CHAR);
    VAR varValue: ARRAY [0..1023] OF CHAR;
  BEGIN (* RetrieveEnvVar *)
    WriteString("SDGetEnvVar("); WriteString(varName); WriteString(") = '");
    SDGetEnvVar(varName,varValue);
    WriteString(varValue); WriteString("'");
    WriteLn;
  END RetrieveEnvVar;
  PROCEDURE RetrieveEnvVarEle (varName: ARRAY OF CHAR; index: INTEGER);
    VAR varValue: ARRAY [0..1023] OF CHAR;
  BEGIN (* RetrieveEnvVarEle *)
    WriteString("SDGetEnvVarListElem("); WriteString(varName); WriteString(","); WriteInt(index,1); WriteString(") = '"
    SDGetEnvVarListElem(varName,index,varValue);
    WriteString(varValue); WriteString("'");
    WriteLn;
  END RetrieveEnvVarEle;
  PROCEDURE RetrieveArgument (argNum: CARDINAL);
    VAR argument: ARRAY [0..1023] OF CHAR;
  BEGIN (* RetrieveArgument *)
    IF argNum<SDGetArgCount() THEN
      WriteString("SDGetArgument("); WriteCard(argNum,1); WriteString(") = '");
      SDGetArgument(argNum,argument);
      WriteString(argument); WriteString("'");
    ELSE
      WriteString("SDGetArgument not called, since argNum "); WriteCard(argNum,1); WriteString(" out of range [0..");
      WriteCard(SDGetArgCount()-1,1); WriteString("]");
    END(*IF*);
    WriteLn;
  END RetrieveArgument;
BEGIN (* AccessEnvironment *)
  INC(testNo);
  WriteLn;
  WriteString("Test "); WriteInt(testNo,1); WriteString(": Accessing the environment"); WriteLn;
  WriteString("==================================="); WriteLn;

  WriteString("SDGetArgCount: returned "); WriteCard(SDGetArgCount(),1);
  WriteString("  => argument index range [0.."); WriteCard(SDGetArgCount()-1,1); WriteString("]");
  WriteLn;
  RetrieveEnvVar("PWD");
  RetrieveEnvVar("HOME");
  RetrieveEnvVar("USER");
  RetrieveEnvVar("PATH");
  RetrieveEnvVarEle("PATH",1);
  RetrieveEnvVarEle("PATH",2);
  RetrieveEnvVarEle("PATH",5);
  (* RASS epc *)
  RetrieveEnvVar(SDM2PathEnvVarName);
  (* P1 *)
  RetrieveEnvVar("M2HOME");
```

```
  RetrieveEnvVar("M2LIB");
  RetrieveEnvVar("M2SYMS");
  RetrieveArgument( 0);
  RetrieveArgument( 1);
  RetrieveArgument( 2);
  RetrieveArgument(12);
  WriteLn;

  (* IF VERSION_ISO *)
  EXCEPT
    WriteLn; WriteString("#### EXCEPTION in AccessEnvironment encountered"); WriteLn;
  (* ENDIF VERSION_ISO *)
END AccessEnvironment;



(*********************************)
(*#####   Interprocessing   #####*)
(*********************************)

(*
  TYPE
    SDLocalMsgID   = INTEGER;
    SDLocalMsgProc = PROCEDURE(ADDRESS);
  PROCEDURE SDSetLocalMsgHandler(    msgID : SDLocalMsgID;     handler : SDLocalMsgProc;
                                 VAR done : BOOLEAN);
  PROCEDURE SDGetLocalMsgHandler(    msgID : SDLocalMsgID; VAR handler : SDLocalMsgProc);
  PROCEDURE SDSendLocalMessage  (msgID : SDLocalMsgID; msg : ADDRESS);

  (* channels are not designed yet. Idea: a channel is a connection between two host
     consisting of two two-way streams:
       command stream: text
       data stream:    binary data with translation facility
  *)
  PROCEDURE SDExecute( progNamePathAndArgs : ARRAY OF CHAR;   (* e.g., "/usr/bin/ls -l" *)
                       VAR progStdOutput   : ARRAY OF CHAR ); (* the program's standard output *)
*)

TYPE
  MyObject = RECORD
              msgID: SDLocalMsgID;
              txt: ARRAY [0..255] OF CHAR;
             END(*RECORD*);

PROCEDURE MyHelloMsgHandler(object: ADDRESS);
  VAR myObj: POINTER TO MyObject;
BEGIN (* MyHelloMsgHandler *)
  WriteString("In MyOtherMsgHandler:  Hello! (object = ");
  WriteHex(object); WriteString(")"); WriteLn;
  WriteString("The object says the msgID is: '");
  myObj := object;
  WriteInt(myObj^.msgID,1); WriteString("'"); WriteLn;
END MyHelloMsgHandler;

PROCEDURE MyOtherMsgHandler(object: ADDRESS);
  VAR myObj: POINTER TO MyObject;
BEGIN (* MyOtherMsgHandler *)
  WriteString("In MyOtherMsgHandler:  How are you? (object = ");
  WriteHex(object); WriteString(")"); WriteLn;
  WriteString("The object contains the text: '");
  myObj := object;
  WriteString(myObj^.txt); WriteString("'"); WriteLn;
END MyOtherMsgHandler;


PROCEDURE ExecuteAnyUnixCmd (VAR testNo: INTEGER);
  CONST asFarAs = 12;
  VAR res: ARRAY [0..4096] OF CHAR; i: CARDINAL; cmd: ARRAY [0..1023] OF CHAR;
    done: BOOLEAN; CONST msgIDHello = 1; msgIDHowAreYou = 2;
  PROCEDURE DocuResult (s: ARRAY OF CHAR; done: BOOLEAN);
  BEGIN (* DocuResult *)
```

```
    WriteString(s); WriteString("  ");
    WriteString("Result of operation: done = ");
    IF done THEN
      WriteString("TRUE");
    ELSE
      WriteString("FALSE");
    END(*IF*);
    WriteLn;
  END DocuResult;
  PROCEDURE SendMessage (msgID: SDLocalMsgID);
    VAR myTransaction: MyObject;
  BEGIN (* SendMessage *)
    WriteString("SDSendLocalMessage: Calling SD MessageHandler with msgID = "); WriteCard(msgID,1); WriteLn;
    myTransaction.msgID := msgID;
    myTransaction.txt := "This is some secret message!";
    SDSendLocalMessage(msgID,ADR(myTransaction));
  END SendMessage;
BEGIN (* ExecuteAnyUnixCmd *)
  INC(testNo);
  WriteLn;
  WriteString("Test "); WriteInt(testNo,1); WriteString("a: Execute Unix commands"); WriteLn;
  WriteString("=============================="); WriteLn;

  res := "";
  cmd := "/bin/ls -l"; (* Note, OS X has no /usr/bin/ls but only /bin/ls *)
  WriteString("Calling Unix command '"); WriteString(cmd); WriteString("':"); WriteLn;
  SDExecute(cmd,res);
  WriteString("SDExecute: returned first "); WriteCard(asFarAs,1); WriteString(" chars: '");
  FOR i:= 0 TO asFarAs DO
    WriteChar(res[i]);
  END(*FOR*);
  WriteString("'"); WriteLn;
  WriteLn;
  WriteString("SDExecute returned res = '");
  WriteString(res);
  WriteString("'"); WriteLn;
  WriteLn;

  WriteLn;
  WriteString("Test "); WriteInt(testNo,1); WriteString("b: Messaging"); WriteLn;
  WriteString("=================="); WriteLn;
  WriteString("SDSetLocalMsgHandler: Installing MyHelloMsgHandler:"); WriteLn;
  SDSetLocalMsgHandler(msgIDHello,MyHelloMsgHandler,done); DocuResult("SDSetLocalMsgHandler",done);
  WriteString("SDSetLocalMsgHandler: Installing MyOtherMsgHandler:"); WriteLn;
  SDSetLocalMsgHandler(msgIDHowAreYou,MyOtherMsgHandler,done); DocuResult("SDSetLocalMsgHandler",done);
  WriteLn;
  WriteString("Sending some messages ..."); WriteLn;
  SendMessage(msgIDHello); SendMessage(msgIDHowAreYou);

  WriteLn;
  (* IF VERSION_ISO *)
  EXCEPT
    WriteLn; WriteString("#### EXCEPTION in ExecuteAnyUnixCmd encountered"); WriteLn;
  (* ENDIF VERSION_ISO *)
END ExecuteAnyUnixCmd;



(*********************************************)
(*#####   Initialization and Termination   ######*)
(*********************************************)

(*
  PROCEDURE SDSetInitProc (    p : PROC; VAR done : BOOLEAN);
  PROCEDURE SDSetTermProc (    p : PROC; VAR done : BOOLEAN);
  PROCEDURE SDExit        (status: INTEGER);
  (* if status < 0, calls HALT after cleaning up *)
  PROCEDURE SDExitMsg     (status: INTEGER; module, procedure, reason : ARRAY OF CHAR);
  (* writes a message composed of module, procedure, reason onto standard
  error and then calls SDEcit *)
*)
```

```
PROCEDURE InitTestProc;
BEGIN (* InitTestProc *)
  WriteString(" --- In InitTestProc --- "); WriteLn;
END InitTestProc;

PROCEDURE TermTestProc;
BEGIN (* TermTestProc *)
  WriteString(" --- In TermTestProc --- "); WriteLn;
  WriteString(" --- This procedure should be called by the run-time system"); WriteLn;
  WriteString("     at the end of each program run, i.e. regardless of crash"); WriteLn;
  WriteString("     or ordinary termination."); WriteLn;
  WriteString(" ---------------------- T H E   E N D ----------------------"); WriteLn;
  WriteLn;
END TermTestProc;

PROCEDURE InitAndTermination (VAR testNo: INTEGER);
  VAR done: BOOLEAN;
  PROCEDURE DocuResult (s: ARRAY OF CHAR; done: BOOLEAN);
  BEGIN (* DocuResult *)
    WriteString(s); WriteString("  ");
    WriteString("Result of operation: done = ");
    IF done THEN
      WriteString("TRUE");
    ELSE
      WriteString("FALSE");
    END(*IF*);
    WriteLn;
  END DocuResult;
BEGIN (* InitAndTermination *)
  INC(testNo);
  WriteLn;
  WriteString("Test "); WriteInt(testNo,1); WriteString(": Initialization and Termination"); WriteLn;
  WriteString("======================================"); WriteLn;
  SDSetInitProc(InitTestProc,done); DocuResult("SDSetInitProc",done);
  SDSetInitProc(InitTestProc,done); DocuResult("SDSetInitProc",done);
  SDSetTermProc(TermTestProc,done); DocuResult("SDSetTermProc",done);
  SDSetTermProc(TermTestProc,done); DocuResult("SDSetTermProc",done);
  WriteLn;
  (* IF VERSION_ISO *)
  EXCEPT
    WriteLn; WriteString("#### EXCEPTION in InitAndTermination encountered"); WriteLn;
  (* ENDIF VERSION_ISO *)
END InitAndTermination;




(*****************************************************************************************)
(*****************************************************************************************)



(**************************)
(*#####   Main Test   #####*)
(**************************)


PROCEDURE DoTheTesting;
  VAR testNo: INTEGER;
BEGIN (* DoTheTesting *)
  testNo := 0;
  DocuTestSysDep;

  OpenAnExistingMacFile(testNo);
  OpenAnExistingUnixFile(testNo);
  OpenAnExistingDOSFile(testNo);
  WriteToANewFile(testNo);
  MakeRandomFileAccess(testNo);
  AlterIOMode(testNo);
  TruncateFile(testNo);
```

```
    RenameDeleteFiles(testNo);
    (*. WriteStdChans(testNo); .*)

    HeapManagement(testNo);

    GetFileAttributesAndClock(testNo);

    DirectoryFunctions(testNo);

    HardwareAndScreen(testNo);

    AccessEnvironment(testNo);

    InitAndTermination(testNo);

    ExecuteAnyUnixCmd(testNo);
  END DoTheTesting;



  (**********************************)
  (******   Module Management   ******)
  (**********************************)

  PROCEDURE InitTestSysDep;
  BEGIN (* InitTestSysDep *)
    uses2ChEOL := (SDEol2<>0C);
  END InitTestSysDep;


BEGIN (* TestSysDep *)
  InitTestSysDep;
  DoTheTesting;
END TestSysDep.
```

D.3 Fliesskommaarithmetik – TstDMFloatEv.mod

Dieses Testprogramm testet eine DM-Implementation auf das Verhalten ihrer Fliesskommazahlen-Arithmetik.

```
MODULE TstDMFloatEv;

  (*
    Used to test DMFloatEnv implementations assuming
    at least all system dependent modules are available

    To test individual exceptions, you best alter the code
    by using either 'halt' or 'dontHalt' (see calls to
    procedure 'DocuAction', flag parameter 'disabled') at the
    begin of each section testing a particular exception
    in test procedure 'GenerateExceptions'.

    Implementation and Revisions:
    ============================

    Author  Date        Description of change
    ------  ----        ---------------------
    AF      14/11/2004  First implementation
  *)


 (* ---------------------------------------------------------------------
    NOTE: This module contains version dependent code, since
    platforms provide different mechanisms to implement some
    of required functions. The code is constructed from this master
    implementation module by editing (commenting and
    uncommenting) locations marked with following conditional
    compilation comments:
```

```
  * VERSION_ISO              Version for any ISO compiler
    VERSION_MacMETH          Version for DM on Macintosh
                             using the MacMETH compiler
    VERSION_EPC              Version for BatchDM on Sun
                             (EPC Modula-2 compiler/library)
    VERSION_P1               Version for P1 compiler on Mac
                             under OS X

  * Generic master version in "SED:Dev:..."
  ------------------------------------------------------------------------ *)


IMPORT SYSTEM;

IMPORT DMFloatEnv;

(* IF VERSION_ISO *)
FROM STextIO IMPORT WriteString, WriteChar, WriteLn;
FROM SWholeIO IMPORT WriteInt, WriteCard;
FROM SRealIO IMPORT WriteFixed;
IMPORT RealMath;
PROCEDURE Exp (x: REAL): REAL; BEGIN RETURN RealMath.exp(x) END Exp;
PROCEDURE Sqrt (x: REAL): REAL; BEGIN RETURN RealMath.sqrt(x) END Sqrt;
PROCEDURE Ln (x: REAL): REAL; BEGIN RETURN RealMath.ln(x) END Ln;
PROCEDURE Entier (x: REAL): INTEGER; BEGIN RETURN RealMath.round(x) END Entier;
(* ENDIF VERSION_ISO *)

(* IF VERSION_MacMETH *) (*.
FROM InOut IMPORT WriteString, WriteLn, WriteInt, WriteCard, WriteReal, Write;
PROCEDURE WriteChar(ch: CHAR); BEGIN Write(ch) END WriteChar;
PROCEDURE WriteFixed(r: REAL; n,fw: CARDINAL); BEGIN WriteReal(r,fw) END WriteFixed;
FROM DMMathLib IMPORT Exp, Sqrt, Ln, Entier;
.*) (* ENDIF VERSION_MacMETH *)

(* IF VERSION_EPC *) (*.
FROM SimpleIO IMPORT WriteString, WriteLn, WriteInt, WriteCard, WriteChar;
FROM RealIO IMPORT WriteReal;
IMPORT MathLib0;
PROCEDURE WriteFixed(r: REAL; n,fw: CARDINAL); BEGIN WriteReal(r,fw,8) END WriteFixed;
PROCEDURE Exp (x: REAL): REAL; BEGIN RETURN MathLib0.exp(x) END Exp;
PROCEDURE Sqrt (x: REAL): REAL; BEGIN RETURN MathLib0.sqrt(x) END Sqrt;
PROCEDURE Ln (x: REAL): REAL; BEGIN RETURN MathLib0.ln(x) END Ln;
PROCEDURE Entier (x: REAL): INTEGER; BEGIN RETURN MathLib0.entier(x) END Entier;
.*) (* ENDIF VERSION_EPC *)


VAR
  debugging: BOOLEAN;

PROCEDURE DocuEnv(s: ARRAY OF CHAR);
  VAR e: DMFloatEnv.FloatEnvironment; b: INTEGER;
BEGIN
  IF NOT debugging THEN RETURN END;
  DMFloatEnv.GetEnvironment(e);
  WriteString("FloatEnvironment: ");
  FOR b:= 0 TO 15 DO
    IF (b=4) OR (b=8) OR (b=12) THEN WriteChar(" ") END;
    IF b IN e THEN WriteChar("L") ELSE WriteChar("0") END;
  END(*FOR*);
  IF s[0]<>0C THEN
    WriteString(" ("); WriteString(s); WriteChar(")");
  END(*IF*);
  WriteLn;
END DocuEnv;

PROCEDURE DocuAnyEnv(e: DMFloatEnv.FloatEnvironment; s: ARRAY OF CHAR);
  VAR b: INTEGER;
BEGIN
  IF NOT debugging THEN RETURN END;
  WriteString("FloatEnvironment: ");
```

```
  FOR b:= 0 TO 15 DO
    IF (b=4) OR (b=8) OR (b=12) THEN WriteChar(" ") END;
    IF b IN e THEN WriteChar("L") ELSE WriteChar("0") END;
  END(*FOR*);
  IF s[0]<>0C THEN
    WriteString(" ("); WriteString(s); WriteChar(")");
  END(*IF*);
  WriteLn;
END DocuAnyEnv;

PROCEDURE DocuAllHalts;
  VAR i: DMFloatEnv.Exception;
BEGIN
  WriteString(" => Halts: ");
  FOR i:= MIN(DMFloatEnv.Exception) TO MAX(DMFloatEnv.Exception) DO
    IF DMFloatEnv.HaltEnabled(i) THEN
      WriteChar("L");
    ELSE
      WriteChar("0");
    END(*IF*);
  END(*FOR*);
  WriteString(" (IUODX) ");
END DocuAllHalts;

PROCEDURE DocuAllExceptions(s: ARRAY OF CHAR);
  VAR i: DMFloatEnv.Exception; e: DMFloatEnv.FloatEnvironment; b: INTEGER;
BEGIN
  WriteString(" => Exceptions: ");
  FOR i:= MIN(DMFloatEnv.Exception) TO MAX(DMFloatEnv.Exception) DO
    IF DMFloatEnv.ExceptionPending(i) THEN
      WriteChar("L");
    ELSE
      WriteChar("0");
    END(*IF*);
  END(*FOR*);
  WriteString(" (IUODX) ");
  IF s[0]<>0C THEN WriteString("  - ") END; WriteString(s);
END DocuAllExceptions;

PROCEDURE DocuAll(s: ARRAY OF CHAR);
BEGIN (* DocuAll *)
  DocuEnv(s);
  DocuAllHalts; DocuAllExceptions(s); WriteLn;
END DocuAll;



CONST
  dontHalt = TRUE; halt = FALSE;

PROCEDURE DocuAction(what: ARRAY OF CHAR; disabled: BOOLEAN; which: DMFloatEnv.Exception);
  CONST test1 = FALSE; (* test1 uses only SetEnvironment *)
BEGIN
  WriteLn;
  WriteString(what);
  IF NOT disabled THEN
    IF test1 THEN
      DMFloatEnv.SetEnvironment(DMFloatEnv.FloatEnvironment{which});
    ELSE
      DMFloatEnv.EnableHalt(which);
    END(*IF*);
    WriteString(" (exception <");
    CASE  which OF
    | DMFloatEnv.invalid: WriteString("invalid");
    | DMFloatEnv.underflow: WriteString("underflow");
    | DMFloatEnv.overflow: WriteString("overflow");
    | DMFloatEnv.divideByZero: WriteString("divideByZero");
    | DMFloatEnv.inexact: WriteString("inexact");
    END(*CASE*);
    WriteString("> ");
    WriteString("should now be intercepted)");
```

```
    WriteLn;
    DocuAllHalts; DocuAllExceptions("before action");
  ELSE
    IF test1 THEN
      DMFloatEnv.SetEnvironment(DMFloatEnv.FloatEnvironment{});
    ELSE
      DMFloatEnv.DisableHalt(which);
    END(*IF*);
    WriteString(" (disabled)"); WriteLn;
    DocuAllHalts; DocuAllExceptions("before action");
  END(*IF*);
  WriteLn;
END DocuAction;


PROCEDURE DocuResult (x: REAL);
  VAR hstr: ARRAY [0..7] OF CHAR;
  PROCEDURE BytesToHexString (x: ARRAY OF SYSTEM.BYTE; VAR hstr: ARRAY OF CHAR);
    TYPE
      Byte = [0..255];
    VAR
      i, ii, n, nb, nh, hihstr : INTEGER; z: Byte;
    VAR  hexDIGIT: ARRAY[0..16] OF CHAR;

    PROCEDURE SetHexDigitsUpperCase( upperC: BOOLEAN );
    BEGIN
      IF upperC THEN  hexDIGIT:= '0123456789ABCDEF';
      ELSE            hexDIGIT:= '0123456789abcdef';
      END(*IF*);
    END SetHexDigitsUpperCase;
  BEGIN
    SetHexDigitsUpperCase(TRUE);
    hihstr:= HIGH(hstr);
    nh:= (hihstr-1) DIV 2;
    nb:= HIGH(x);
    IF (nh < nb) THEN  n:= nh;
    ELSE               n:= nb;
    END(*IF*);
    ii:=  0;
    FOR i:= 0 TO n DO
      z := Byte(x[i]);
      hstr[ii]:= hexDIGIT[z DIV 16];  INC( ii );
      hstr[ii]:= hexDIGIT[z MOD 16];  INC( ii );
    END(*FOR*);
    IF (ii <= hihstr) THEN  hstr[ii]:= 0C;  END(*IF*);
  END BytesToHexString;

BEGIN (* DocuResult *)
  WriteString("result = "); WriteFixed(x,8,15);
  WriteString("  = ");
  BytesToHexString(x,hstr);
  WriteString(hstr);
  WriteLn;
END DocuResult;


PROCEDURE DocuPostCondition(exceptionToClear: DMFloatEnv.Exception);
BEGIN (* DocuPostCondition *)
  DocuEnv("after  action");
  DocuAllHalts; DocuAllExceptions("after action"); WriteLn;
  DMFloatEnv.ClearException(exceptionToClear);
  (* since several exceptions trigger also inexact, clear it always *)
  DMFloatEnv.ClearException(DMFloatEnv.inexact);
  DocuEnv("after  clearing");
  DocuAllHalts; DocuAllExceptions("after clearing"); WriteLn;
END DocuPostCondition;
```

```
PROCEDURE ShowFloatEnvironment;
BEGIN (* ShowFloatEnvironment *)
  debugging := FALSE;
  WriteLn;
  WriteString("Current Float environment, halts, and exceptions: "); WriteLn;
  DocuAll("");
  WriteLn;
  WriteString("Legend: "); WriteLn;
  WriteChar(" "); WriteInt(DMFloatEnv.invalid,0); WriteString(" = I  invalid"); WriteLn;
  WriteChar(" "); WriteInt(DMFloatEnv.underflow,0); WriteString(" = U  underflow"); WriteLn;
  WriteChar(" "); WriteInt(DMFloatEnv.overflow,0); WriteString(" = O  overflow"); WriteLn;
  WriteChar(" "); WriteInt(DMFloatEnv.divideByZero,0); WriteString(" = D  divideByZero"); WriteLn;
  WriteChar(" "); WriteInt(DMFloatEnv.inexact,0); WriteString(" = X  inexact"); WriteLn;
  WriteLn;
  WriteLn;
END ShowFloatEnvironment;



PROCEDURE GenerateExceptions;
  CONST false = 0; true = 1; adhoc = 2; all = 0; none = 0;
  VAR
    environ: DMFloatEnv.FloatEnvironment;
    a,b,c,d,e,f,g,h,i,x,y,z: REAL; ok,skip: BOOLEAN; xl,yl: LONGREAL;
    int: INTEGER;
    setIEEEdefaultHalting: INTEGER;

    ch: CHAR;
BEGIN (* GenerateExceptions *)

  debugging := FALSE;
  DMFloatEnv.SetEnvironment(DMFloatEnv.IEEEFloatDefaultEnv);
  DocuAll("IEEE default");
  DMFloatEnv.SetEnvironment(DMFloatEnv.DMFloatDefaultEnv);
  DocuAll("DM default");
  WriteLn;
  WriteLn;

  (* General mode settings actually ineffective, if individual halt
  enabling, disabling is fully effective *)
  debugging := TRUE;
  IF debugging THEN
    setIEEEdefaultHalting := true;
    CASE setIEEEdefaultHalting OF
    | true:
      WriteString("### Mode where nothing should be intercepted"); WriteLn;
      DocuEnv("old");
      DMFloatEnv.SetEnvironment(DMFloatEnv.IEEEFloatDefaultEnv);
      DocuAll("IEEE default");
    | false:
      WriteString("### Mode where 'invalid, overflow, div by 0' should be intercepted"); WriteLn;
      DocuEnv("old");
      DMFloatEnv.SetEnvironment(DMFloatEnv.DMFloatDefaultEnv);
      DocuAll("DM default");
    ELSE
      WriteString("### Mode where ad hoc exceptions should be intercepted"); WriteLn;
      DocuEnv("old");
      DMFloatEnv.SetEnvironment(DMFloatEnv.FloatEnvironment{DMFloatEnv.haltIfOverflow});
      DocuAll("after setting adhoc");
    END(*CASE*);
  END(*IF*);


  debugging := FALSE;


  DocuAction("Signaling NaN",dontHalt,none);
  x  := REAL(0FFB00000H);
  y  := REAL(0FFA0FFFFH);
  z  := x + y;
  DocuPostCondition(all);
```

```
DocuAction("Unordered Condition (= invalid) ",dontHalt,DMFloatEnv.invalid);
a  := REAL(07FFFFFFH);
b  := REAL(0FFFFFFFFH);
ok := (a > b);
DocuPostCondition(DMFloatEnv.invalid);


DocuAction("Underflow",dontHalt,DMFloatEnv.underflow);
f  := REAL(080000001H);
d  := f + f;
DocuResult(d);
DocuPostCondition(DMFloatEnv.underflow);


DocuAction("Overflow",dontHalt,DMFloatEnv.overflow);
i  := MAX(REAL);
e  := i * i;
DocuResult(e);
DocuPostCondition(DMFloatEnv.overflow);


(* The P1 compiler intercepts division by zero always, due to
a previous bug in the Carbon library. To successfully control
this behavior you need to suppress the code generation for this
test *)
(* IF VERSION_P1 *)
<* PUSH *>
<* ASSIGN ( DivZeroCheck , FALSE ) *>
(* ENDIF VERSION_P1 *)
DocuAction("Zero Divide",dontHalt,DMFloatEnv.divideByZero);
d  := 1.0; e := 0.0;
c  := d / e;
DocuResult(c);
DocuPostCondition(DMFloatEnv.divideByZero);
(* IF VERSION_P1 *)
<* POP *>
(* ENDIF VERSION_P1 *)


DocuAction("Inexact",dontHalt,DMFloatEnv.inexact);
(* IF VERSION_MacMETH *) (*.
x := 1.0E-35;
y := 1.0E-10 * x;
.*) (* ENDIF VERSION_MacMETH *)
(* IF VERSION_EPC *) (*.
x := 1.0E-35;
y := 1.0E-10 * x;
.*) (* ENDIF VERSION_EPC *)
(* IF VERSION_P1 *)
xl := 1.0E-300;
yl := 1.0E+300 * xl;
(* ENDIF VERSION_P1 *)
DocuPostCondition(DMFloatEnv.inexact);


(* Testing exceptions in math functions *)

DocuAction("Operand Error, i.e. Sqrt(-1)",dontHalt,DMFloatEnv.invalid);
g  := -1.0;
a  := Sqrt(g);
DocuResult(a);
DocuPostCondition(all);

DocuAction("Operand Error, i.e. Ln(-1)",dontHalt,DMFloatEnv.invalid);
g  := -1.0;
b  := Ln(g);
DocuResult(b);
DocuPostCondition(all);
```

```
    (* IF VERSION_P1 *)
    <* PUSH *>
    <* ASSIGN ( OverflowCheck , FALSE ) *>
    (* ENDIF VERSION_P1 *)
    DocuAction("Integer overflow, i.e. Entier(MAX(REAL))",dontHalt,DMFloatEnv.overflow);
    g  := MAX(REAL);
    int:= Entier(g);
    DocuPostCondition(all);
    (* IF VERSION_P1 *)
    <* POP *>
    (* ENDIF VERSION_P1 *)

    DocuAction("Overflow, i.e. Exp(MAX(REAL))",dontHalt,DMFloatEnv.overflow);
    h  := MAX(REAL);
    i  := Exp(h);
    DocuResult(i);
    DocuPostCondition(all);

  END GenerateExceptions;



BEGIN (* TstDMFloatEv *)
  WriteLn;
  WriteString("At beging of program 'TstDMFloatEv'"); WriteLn;
  WriteString("==================================="); WriteLn;
  ShowFloatEnvironment;
  GenerateExceptions;
END TstDMFloatEv.
```