

FEAP - Installations-Manual

Stand: 17.03.2015

Inhaltsverzeichnis

1 Voraussetzungen	3
2 Vorgehensweise beim Update von Visual Studio bzw. eines Compilers	3
3 Vorgehensweise beim Update eines Compilers	3
4 Installation MS Visual Studio 2010 Professional	4
5 Installation von Intel(©) Visual Studio for IA-32 and IA-64	6
6 Installation von Silverfrost(©) FTN95 for IA-32	11
7 Installation von FEAP	15
8 Projekt FEAP-Int in MS Visual Studio 2010 anlegen (IA32+IA64)	20
9 Projekt FEAP-FTN in MS Visual Studio 2010 anlegen (IA32)	38
10 FEAP-FTN (IA32) mit Makefile in Ultredit	45
11 Anwendung der Debugger	54
12 Anwendung des Manuals	56
13 Anwendung von Parallellösern	56
14 offene Probleme und Infos	57
15 Erstellung von Studenten-Versionen	57
16 Verwendung	59
17 FAQ - Installationsprobleme	59
18 FEAP Shared Memory Dokumentation-Manual	60
18.1 Einleitung	60
18.2 Voraussetzungen	60
18.3 Einrichtung	60

18.4 Benutzung	60
19 FEAP Shared Memory Dokumentation-Anhang	62
19.1 Modifizierte Quelldateien	62
19.2 Known Issues - Probleme und Einschränkungen	62
19.3 Fehlermeldungen und Lösungsvorschläge	64
19.4 Fragen und Antworten	65

1 Voraussetzungen

1. Betriebssystem

- Windows 7-32 oder Windows 7-64
- [(Windows 8 -INTEL Fortran Studio soll gehen)]

2. Projektumgebung

- MS Visual Studio 2010 SP1 Professional
- [MS Visual Studio 2012 Professional - INTEL Fortran Studio soll gehen]

3. Compiler

- Intel Visual Fortran Studio XE 2013 Compiler-Version 13.1.0.149 Update 4
- Intel Visual Fortran Studio XE 2015 Compiler-Version 15.0.1.148 Update 1
- Silverfrost FTN95 6.35 [FTN95 6.00]

2 Vorgehensweise beim Update von Visual Studio bzw. eines Compilers

1. Compiler(INTEL, FTN95) deinstallieren-alle Teil-und Zusatzprogramme!!
2. Visual Studio ALT deinstallieren.
3. diverse Zusatzprogramme zu Visual Studio ALT deinstallieren,
vgl. Systemsteuerung/Software/Programme
4. Restart Rechner.
5. Visual Studio NEU installieren.
6. Compiler(INTEL, FTN95) in Visual Studio hinein installieren.
7. nach jeweiligem FEAP-Projekt (Endung INTEL=.sln/.vfproj, FTN95=.sln/.ftn95p)
suchen. Dies sollte gefunden und automatisch konvertiert werden. Ggf. muss noch alles
neu compiliert werden.

3 Vorgehensweise beim Update eines Compilers

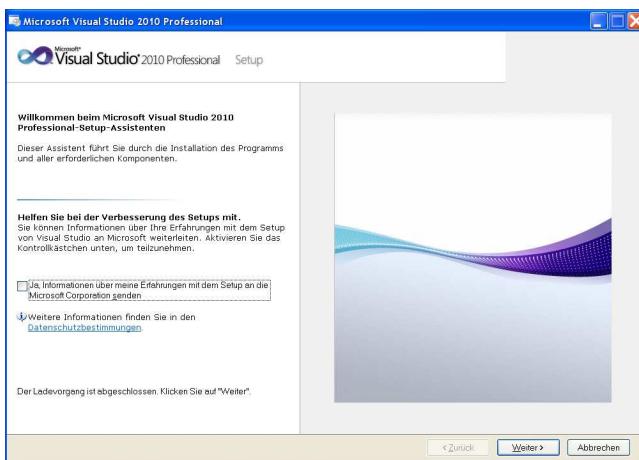
1. Compiler(INTEL, FTN95) deinstallieren.
2. Compiler(INTEL, FTN95) in Visual Studio hinein installieren.
3. Projekt muss neu compiliert werden.

Hinweis: Es können mehrere INTEL-Compiler gleichzeitig installiert sein.

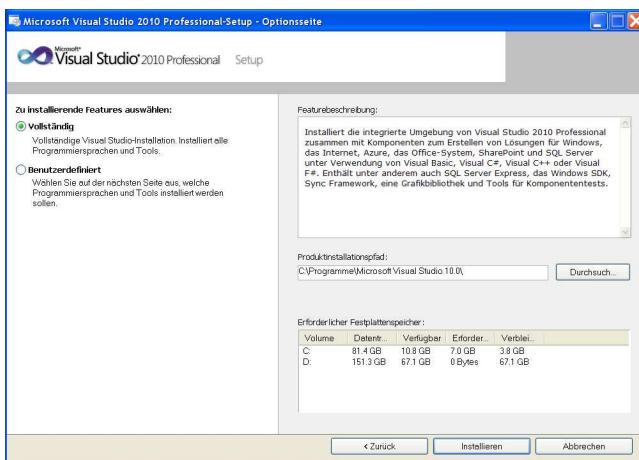
4 Installation MS Visual Studio 2010 Professional

Hinweis: Alle Bilder und Menüs beziehen sich auf MS Visual Studio 2010 Professional. Daher kann das aktuelle Aussehen bzw. die Inhalte einzelner Menüs bei Verwendung anderer Versionen von MS Visual Studio hiervon abweichen!

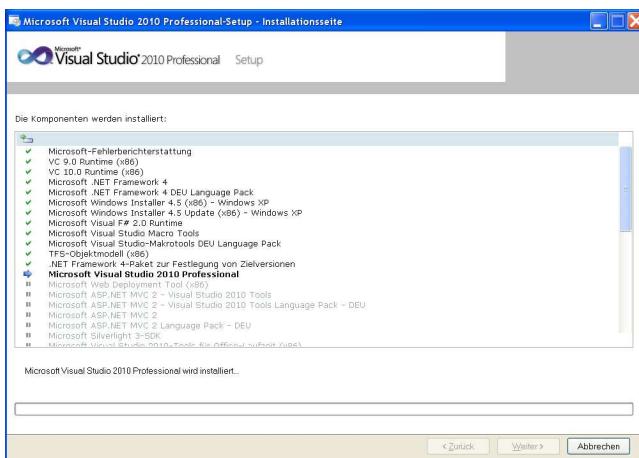
1. Start



2. vollständig, vorgeschlagenen Pfad beibehalten



3. Installation



4. Microsoft-Update laufen lassen, ggf. temporär installieren, um aktuelle Version zu erhalten-das kann dann auch mehrmals einen Update erfordern. Stand 11/2012 MS VS10 SP1

The screenshot shows the Microsoft Update interface. On the left, there's a sidebar with links like 'Microsoft Update - Startseite', 'Optionen' (selected), 'Updateverlauf anzeigen', 'Ausgeblendete Updates wiederherstellen', 'Einstellungen ändern', 'FAQ (Häufig gestellte Fragen)', 'Hilfe und Support', 'Administrator-Optionen', and 'Hardware support suchen'. The main area has a title 'Installationsergebnisse anzeigen' and a message 'Jetzt neu starten, um die Installation der Updates abzuschließen' (Restart now to finish the update installation). Below that is a section titled 'Installationsübersicht' with a table:

Erfolgreich:	7
Fehler:	0
Weitere Updates:	0

Under 'Erfolgreiche Updates', there are two sections: 'Microsoft Windows XP' and 'Microsoft Visual Studio 2010'. Each section lists several security updates with their respective KB numbers.

Microsoft Windows XP
Sicherheitsupdate für Microsoft .NET Framework 4 unter Windows XP, Windows Server 2003, Windows Vista, Windows 7, Windows Server 2008 x86 (KB2487367)
Sicherheitsupdate für Microsoft .NET Framework 4 unter Windows XP, Windows Server 2003, Windows Vista, Windows 7, Windows Server 2008 x86 (KB2656351)
Microsoft Visual Studio 2010
Sicherheitsupdate für Microsoft Visual Studio 2010 XML-Editor (KB2251489)
Sicherheitsupdate für Microsoft Visual Studio 2010 (KB2542054)
Sicherheitsupdate für Microsoft Visual Studio 2010 (KB2644980)

5. Visual Studio starten:

Allgemeine Entwicklungsumgebung für Fortran vorgeben.

5 Installation von Intel(©) Visual Studio for IA-32 and IA-64

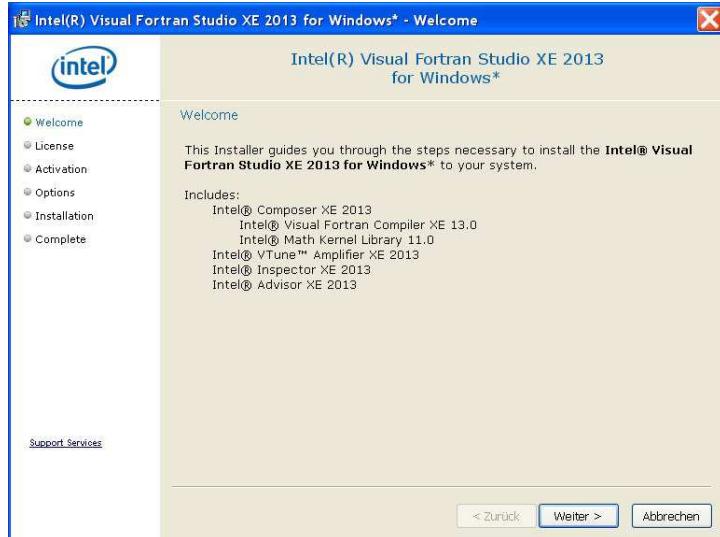
Hinweise:

- Soll auch der Silverfrost(©) FTN95-Compiler in Visual Studio installiert werden, gibt es möglicherweise ein Reihenfolgeproblem. Installations-Reihenfolge INTEL/FTN95 führte zu einem instabilen Visual Studio(Absturz) für FTN95-Projekte, Installations-Reihenfolge FTN95/INTEL funktioniert.
- Alle Bilder und Menüs beziehen sich auf Intel(©) Intel Visual Studio XE 2013 Compiler-Version 13.0.0.089

1. Installiere Intel(©) Visual Studio for IA-32 and IA-64

f_studio_xe_2013_novshell_setup.exe

(a) Übersicht

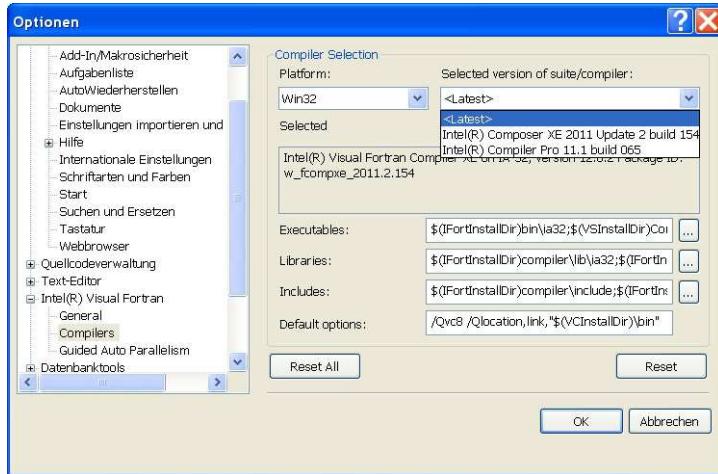


(b) Infos zur Parallelinstallation



Es können mehrere Intel-Compiler parallel installiert und verwendet werden. Diese Meldung hier schließt die alte Version 10 aus.

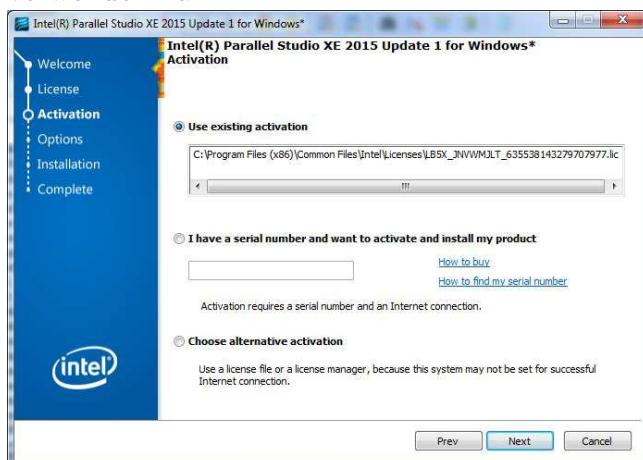
Die Einstellung erfolgt dann in Visual Studio unter Extras>Optionen



(c) Choose alternative activation!!!



Das sieht mittlerweile so aus, so dass man evtl. die vorhandene Lizenz weiter verwenden kann:



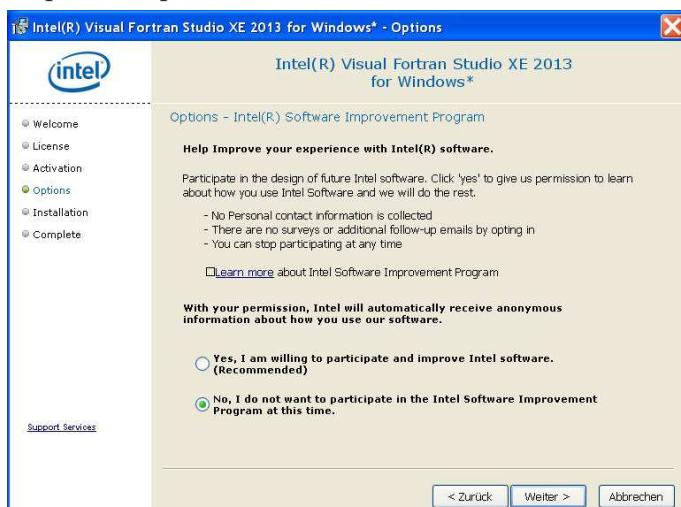
(d) Use a License file



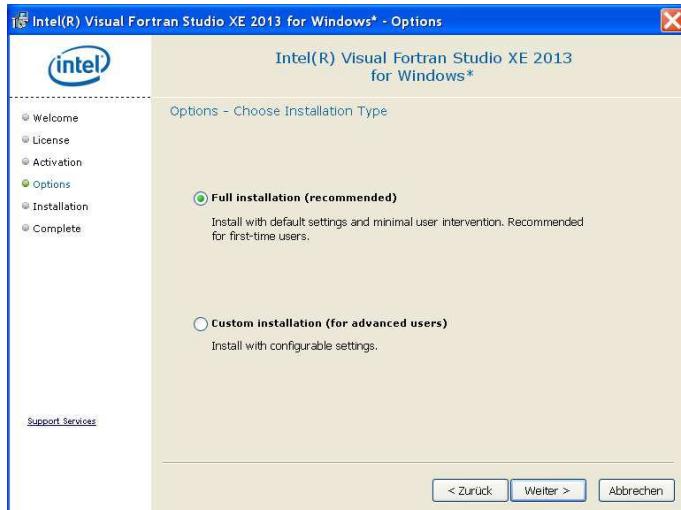
- (e) vorhandene Lizenzierte-Key-file auf dem Rechner deponieren und dann hier Pfad angeben.



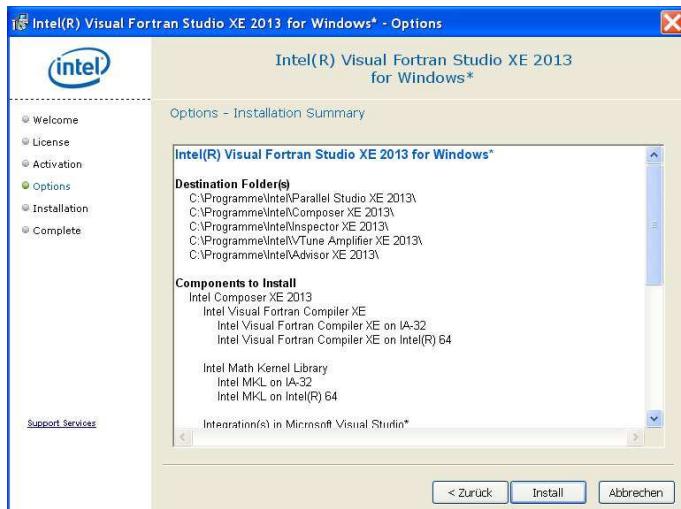
- (f) Improve experience with INTEL software



(g) Full Installation



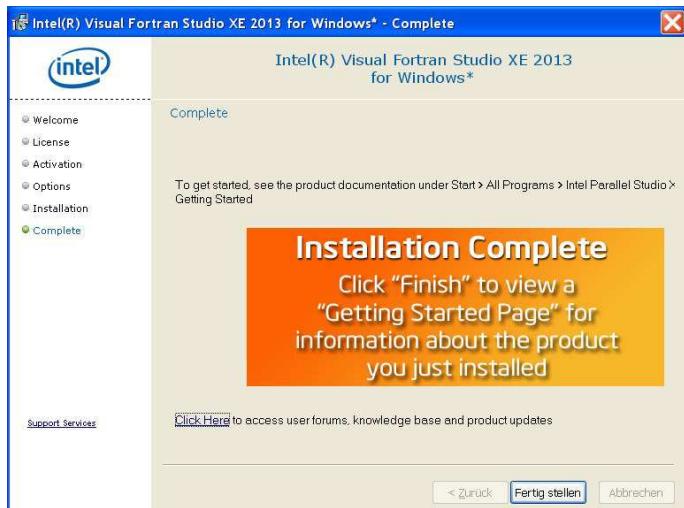
(h) Installationsübersicht



(i) Installation



(j) Installationsabschluss-keine Registrierung!



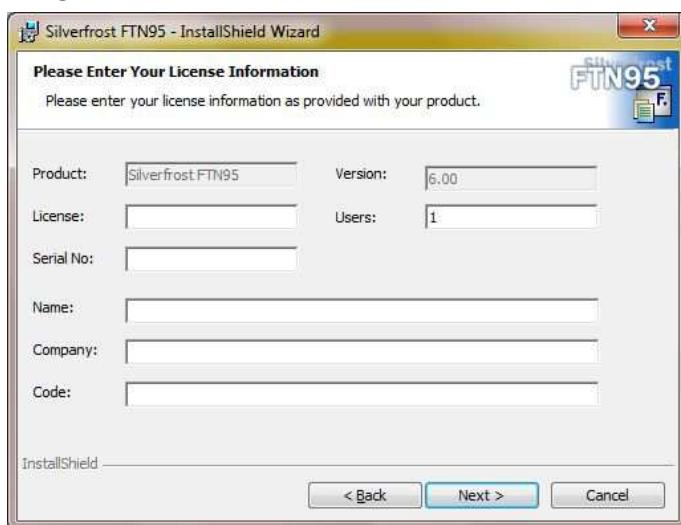
6 Installation von Silverfrost(©) FTN95 for IA-32

1. Vorbemerkung: Die Installation soll mit Einbindung in Visual Studio erfolgen. Das zugehörige FEAP-Projekt ist in Abschnitt 9 beschrieben. Alternativ kann auch eine Verwendung ohne Einbindung in Visual Studio erfolgen. Dann muss das Handling von FEAP mit Batch-Files erfolgen, siehe Abschnitt 10.
2. Installiere Silverfrost(©) FTN95 for IA-32
`ftn95_full.exe`

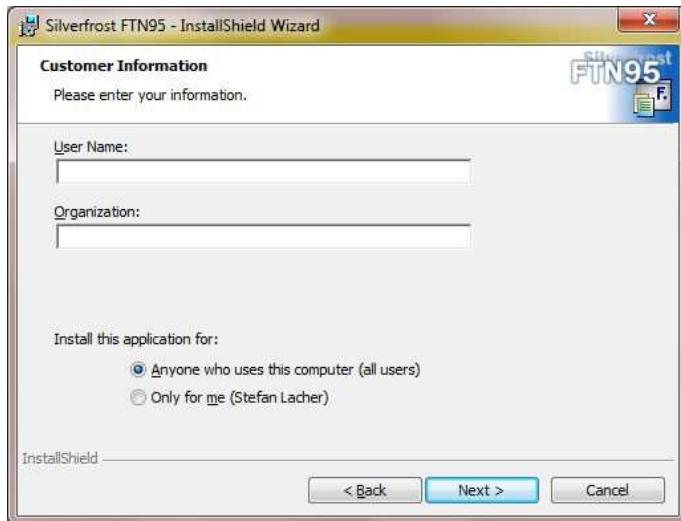
(a) Installationsfenster



(b) Eingabe der Lizenzdaten



(c) Eingabe Benutzerdaten

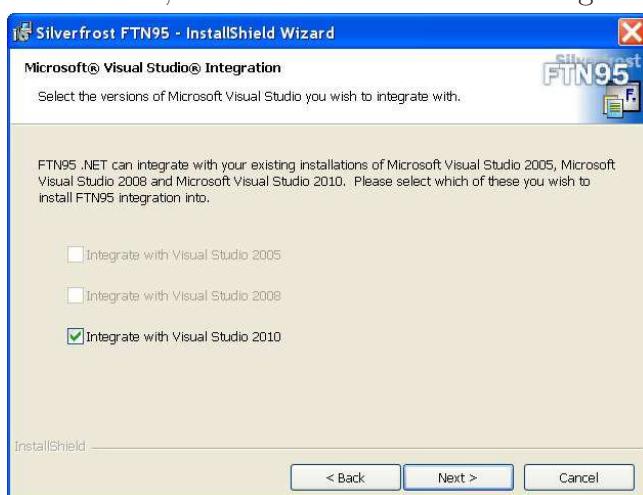


(d) Eingabe Installationsumfang und -pfad - c:\programme\ftn95

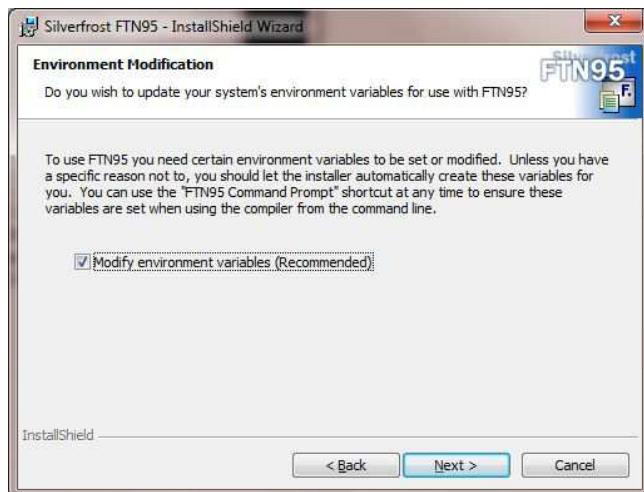


(e) Einbindung in Visual Studio

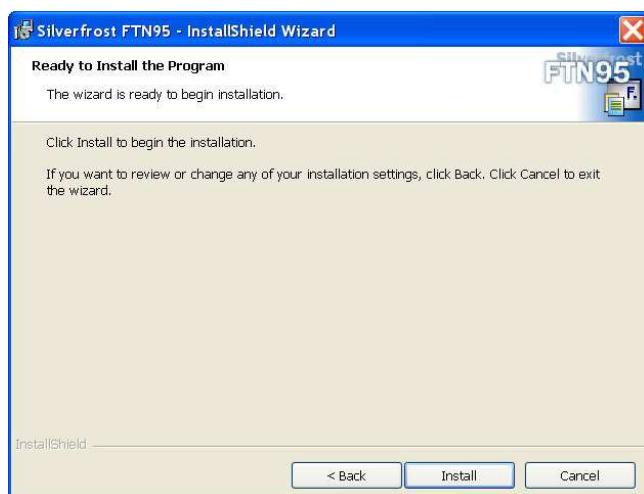
Wenn nicht, kann nur mit Batch-Dateien gearbeitet werden.



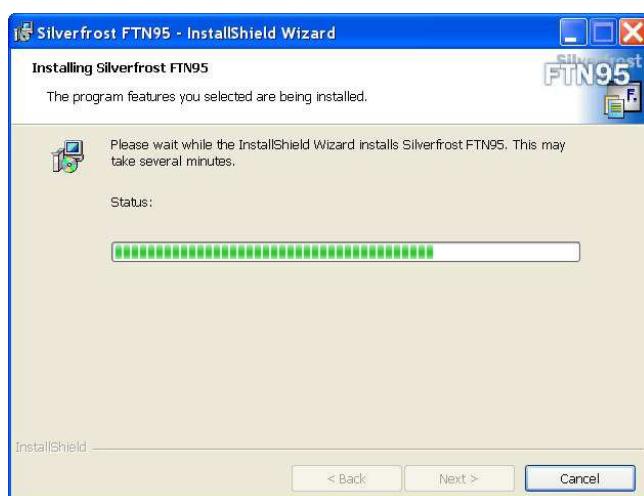
(f) Setzen der Umgebungsvariablen

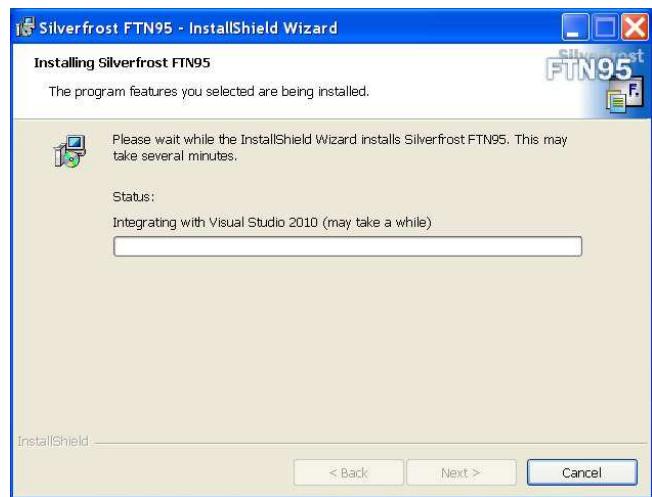


(g) Bereit zur Installation



(h) Installation





7 Installation von FEAP

1. FEAP-System-Installationen

- (a) DLL-Datei cal.dll nach C:\WINDOWS\System kopieren.
- (b) Verzeichnisse "FEAP", "FEAP\add", und "FEAP\icons" anlegen.
- (c) Umgebungsvariable "FCFG=path\FEAP\add\" definieren, (mit letztem \)!.
- (d) Datei feap_registration_key.txt in Verzeichnis FEAP\add kopieren.
- (e) Datei feapwin.ini in Verzeichnis FEAP\add kopieren.

2. FEAP-Umgebungsprogramme

- (a) in der Datei feapwin.ini Pfade zu den FEAP-Umgebungsprogrammen anlegen.
Bei Leerzeichen Anführungszeichen verwenden, siehe unten beim Acrobat Reader.
Beispiel:

```
feapsal Position von FEAP - SALFORD
notwendig für die Hilfsprogramme und Aufruf FE$^2$
path\feap95\feap.exe
```

```
feapint Position von FEAP-INTEL(mit Standard-Graphics)
notwendig für Aufruf des Micro-Problems bei FE$^2$
path\feap95\intel\feap\debug-s\feap.exe
```

```
feaped Position des Editors
path\FEAP\add\feap_ed.exe
```

```
fnege Position des Netzgenerierungsprogrammes NEGE
path\FEAP\add\negef.exe
```

```
fgmesh Position des Netzgenerierungsprogrammes GMESH
path\FEAP\add\gmesh.exe
```

```
fproce Position des Prozedureeditors
path\FEAP\add\f_proced.exe
```

```
fadobe Position ADOBE
c:\Program Files (x86)\Adobe\Acrobat 10.0\Acrobat\AcroRd32.exe
```

```
manual Position des Hilfe-Files
path\FEAP\add\feap.hlp
```

```
editor Position eines Editors UEDIT, Notepad etc.
c:\Program Files (x86)\UltraEdit\uedit32.exe
```

```
psview Position von Ghostview
```

```

c:\Program Files (x86)\gs\gsview\gsview\gsview32.exe

intnet Internet-Explorer
start iexplore.exe

fcylt ylt-netzgenerierer
path\FEAP\add\cylt.exe

wininp Name/Position/Größe Dialog-Fenster
F E A P   Dialog   Window
0.60,0,0.40,0.95

wingra Name/Position/Größe Graphik-Fenster
F E A P   Graphic   Window
0.0,0.0,0.595,0.72

winhlp Name/Position/Größe Hilfe-Fenster
F E A P   Help   Window
0.0,0.725,0.60,0.23

Bildschirmgröße 0<1_x<1  0<1_z<1 x-rechts, z-unten, 0,0 = links oben
Parameter 1/2 = x_0, z_0 des Fensters
Parameter 3/4 = l_x, l_z des Fensters

performance bar nur für SALFORD: zeigen ab Knoten-/Elementzahl
2000000

tplopt Anzahl der Punkte für TPL0
1000

end

```

Hinweise:

- Die FEAP-spezifischen Pfade (hier FEAP\add) können auch beliebig einzeln anders gesetzt werden.
- Die Position des Acrobat Readers kann automatisch gesucht werden, wenn nach 'fadobe' eine Leerzeile eingefügt wird. Beim ersten Aufruf wird die Position gesucht und in den ini-File kopiert.
- Die Eingabereihenfolge ist beliebig.
- überflüssige Angaben sind optional, d.h. können weggelassen werden: z.B. winhlp, wininp.

(b) zugehörige Dateien an die passenden Stellen kopieren (hier FEAP\add)

```

FEAP_ED.exe,
NEGEF.exe,
GMESH.exe,

```

CYLT.exe,
F PROCED.exe,
FEAP.hlp,
GotoAcro.exe,
feap_ass.exe

- (c) FEAP-Manual installieren
Datei FEAP-MAN.pdf in das Verzeichnis FEAP\add kopieren.
- (d) FEAP-File-Eingabe für Windows/Intel installieren
Datei Filnamw_i.exe in das Verzeichnis FEAP\add kopieren.

3. FEAP-Quellprogramme

- (a) Verwendete Fortran-Quellen-für beide Compiler

arclen.for
bfgs.for
contact.for
cor.for
crit.for
csr_help.for
curve.for
cvuser.for
cylt1.for
cylt2.for
cylt3.for
cyltalg.for
dalloc.for ##new
dplot.for
dynamics.for
eig1.for
elmlib.for
elmt01s.for
....
elmttnns.for
ext.for
feap.for
feap_micro.for
feapcfg.for
feaps1.for
feaps2.for
feaps3.for
feaps4.for
feaps5.for
feaps6.for
fepost.for

gframe.for
help.for
hpgl.for
iniuser.for
isogeo.for
lanczos.for
mat.for
mate3d01.for
mate3d02.for
mate3d03.for
mate3d04.for
mate3d05.for
mate3d06.for
mate3d07.for
mate3d08.for
mate3d09.for
mate3d10.for
mate3d11.for
mate3d12.for
mate3d14.for
mate3d15.for
matlib3d.for
module.for ##new
paraview.for
pardisoint.for
pbcg.for
pgmres.for
pgmres2.for
plot.for
pmacr.for
pmesh1.for
pofeap.for
pplotf.for
precond.for
reme_nl1.for
reme_nl2.for
remesh1.for
remesh2.for
remesh3.for
section.for
simplex_qld.for
smoo_control.for
smsolv.for
subsp.for
superluint.for

```
tecplot.for  
ueigen.for  
unika.for
```

(b) **Verwendete Include-Files**

--

(c) **Verwendete Resource-Files**

```
feapico.rc
```

Die Pfade in der Datei `feapico.rc` müssen angepasst werden!

(d) **Zusätzliche Quellen-INTEL**

```
allocation_int.for ##new  
plot_intel.for  
plot_intel_cw.for  
feap_shmem.for  
feast.for
```

(e) **Zusätzliche Quellen-FTN95**

```
allocation_sal.for ##new  
plot_sal.for  
plot_sal_cw.for
```

(f) **Zusätzliche Bibliotheken-FTN95**

SuperLU.lib

(g) **ICON-Dateien**

Kopieren der folgenden Dateien nach FEAP\icons
*.ico, *.bmp aus dem Verzeichnis icons

(h) **Eigene Dateien/Elemente**

Kopieren der eigenen Elemente nach FEAP und Anpassen der Datei elmlib.for an die eigenen Elemente.

8 Projekt FEAP-Int in MS Visual Studio 2010 anlegen (IA32+IA64)

Hinweise: Alle Bilder und Menüs beziehen sich auf MS Visual Studio 2010 Professional und frühere Versionen. Das aktuelle Aussehen bzw. die Inhalte einzelner Menüs kann ggf. hiervon abweichen!

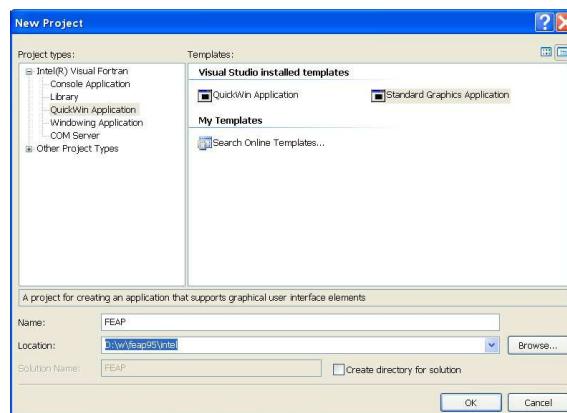
Bei Neuinstallation von Visual Studio nach dem FEAP-Projekt (Endung .sln) suchen. Dies sollte gefunden und automatisch konvertiert werden. Ggf. muss noch alles neu compiliert werden.

- **FEAP-Quellen einbinden**

1. MS Visual Studio 2010 starten.
2. Datei>Projekt eröffnen: **File>New>project**

Type: Intel(R) Visual Fortran quickwin application
und Template: standard graphics application.

Der anzugebende Projektpfad ist nicht der Pfad zu den FEAP-Quellen, sondern der Pfad, wo später die Datei **FEAP.exe** sowie alle projektspezifischen Angaben liegen. Vorschlag: **FEAP\intel**.



3. Projektmappenkonfiguration: **Debug**
4. Projektmappenplattform:
 - WIN32
 - WIN64

Neue Plattformen werden mit dem Konfigurationsmanager (im Fenster Projektmappenplattformen) angelegt.

FEAP wird dann je nach aktiver Plattform als 32-Bit oder 64-Bit Version erstellt. Bei Start aus dem MS Visual Studio wird die aktive Version genommen. Falls mit einem Batch-File gestartet wird, ist darauf zu achten, das die 'richtige' FEAP-Version genommen wird. Es existiert je Plattform eine EXE-Datei!

5. Fortran(source)-Quellen laden:
Im **Solution Explorer** (rechts) **FEAP Source Files** anklicken, dann mit

Project>Add Existing Items alle für INTEL erforderlichen Fortran-Quellen in das Projekt laden. Die **Include-Dateien *.h** sind nicht zu laden!

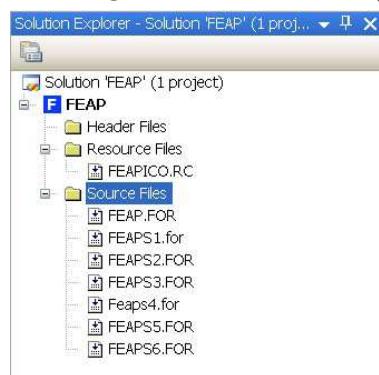
Ggf. im **Solution Explorer** die FTN95-Dateien mit der rechten Maustaste anklicken und aus dem Projekt löschen.

6. Resource-Quellen laden:

Im **Solution Explorer** (rechts) **FEAP Resource Files** anklicken, dann mit **Project>Add Existing Items** die Datei **feapico.rc** in das Projekt laden.

Doppelklick auf die Datei **feapico.rc** im **Solution Explorer** öffnet diese Datei. Hierin sind die angegebenen Pfade zu überprüfen und ggf. anzupassen.

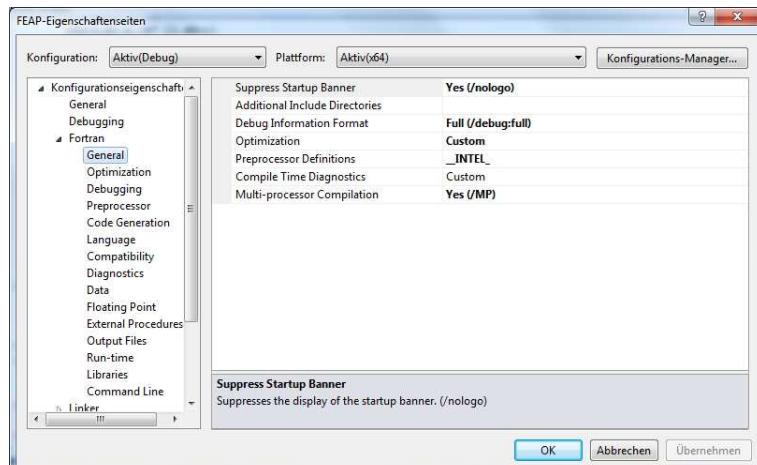
Das Ergebnis sieht so aus (Teilbeispiel!):



• FEAP-Projekteigenschaften (Compilier-Optionen-DEBUG)

1. Im **Solution Explorer** (rechts) **FEAP** anklicken, dann **Project>properties** anklicken.
Compilier- und Linkeigenschaften in den Untermenüs einstellen:
2. Fortran-Compiler

01-General



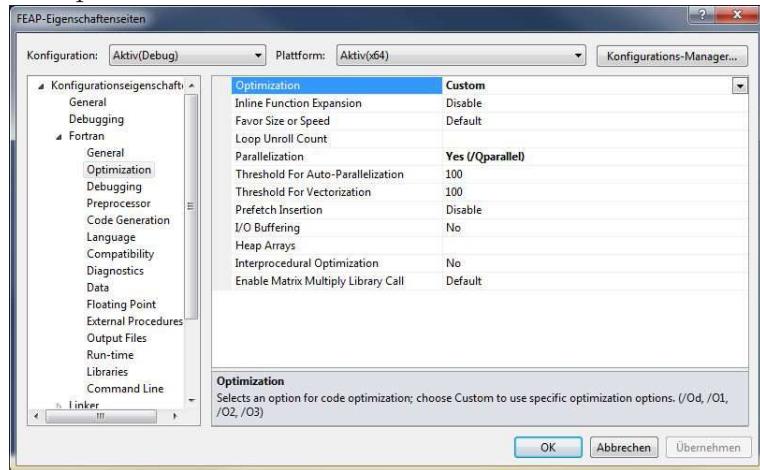
Debug Information Format: Full(/debug:full)

Optimization: Custom

Preprocessor Definitions: __INTEL__

Multi-processor Compilation: Yes(/MP)

02-Optimization



Hier sind verschiedene Optimierungs-Levels möglich:

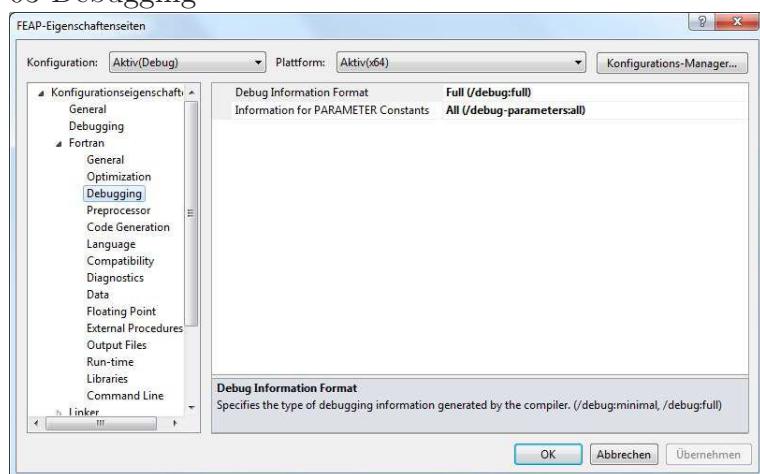
- Basis(sicher): Optimization: Disable (/Od)
- Schnell: Optimization: Custom
- Super-Schnell:
 Optimization: Maximize Speed plus Higher Level Optimizations (/O3)

Bei Optimierung geht die Compilierzeit deutlich hoch! Der Compiler vektorisiert Schleifen. Der PARDISO-Löser ist schon optimiert, daher gibt es dort keinen Zeitgewinn!

weiter zu setzen:

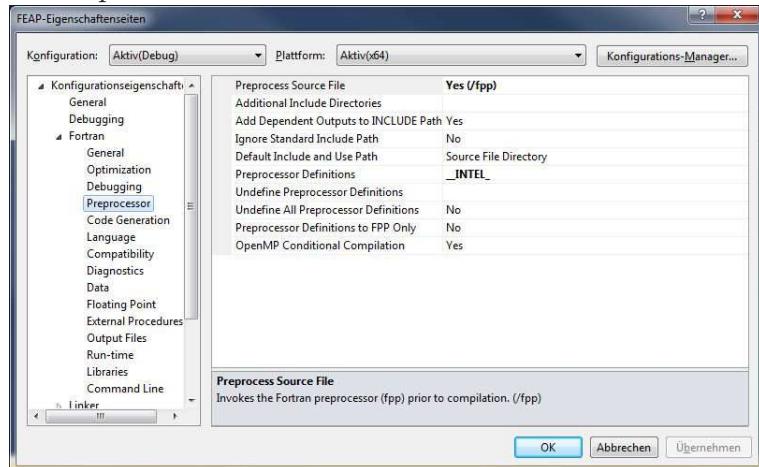
Parallelization: Yes

03-Debugging



Zum Zugriff auf alle Parameter muss hier Full und All gesetzt werden.

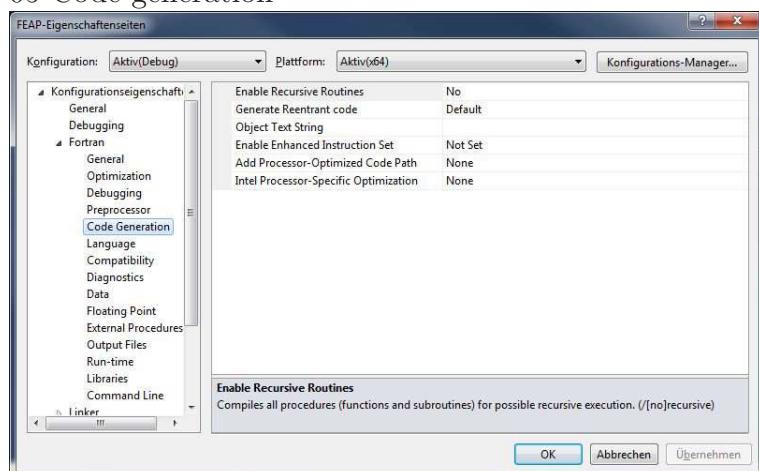
04-Preprocessor



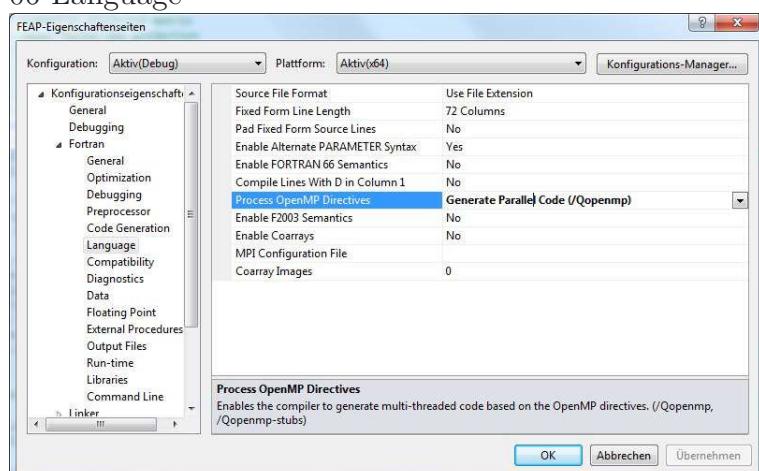
Preprocess Source File: Yes (/fpp)

Preprocessor Definitions: __INTEL__

05-Code generation



06-Language

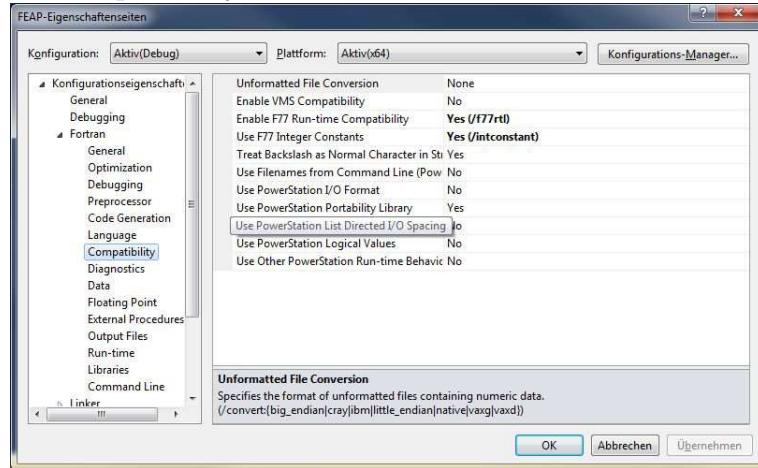


Process OpenMP Directives: Generate Parallel Code (/Qopenmp)

Elementschleife in FEAP läuft damit parallel.

Für ein serielles Debuggen kann die Option temporär entfernt werden.

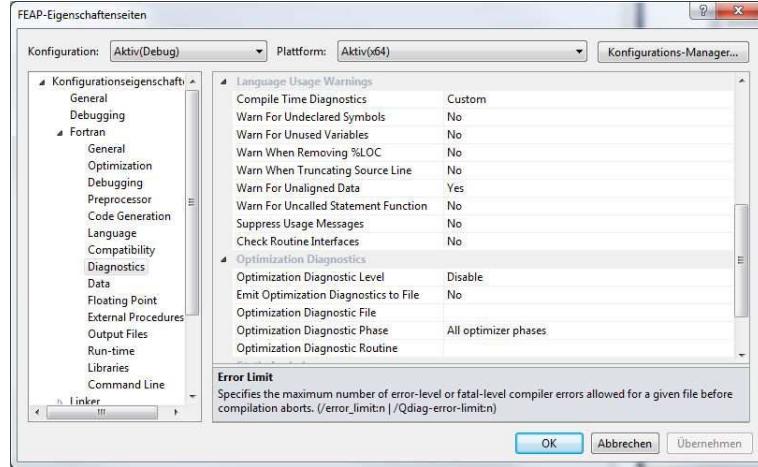
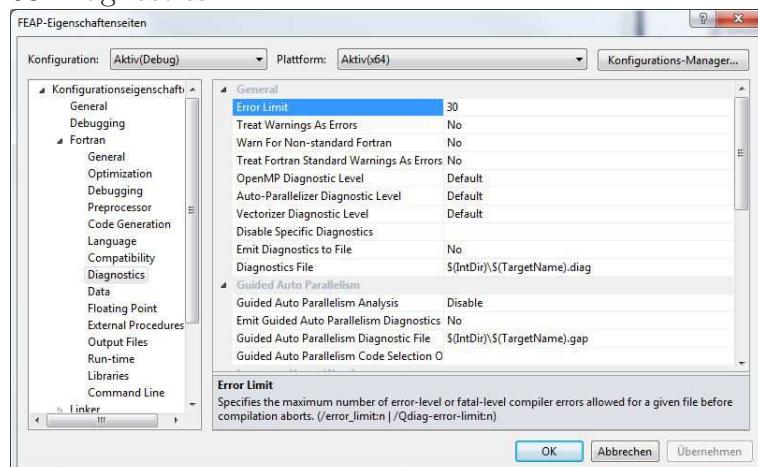
07-Compatibility



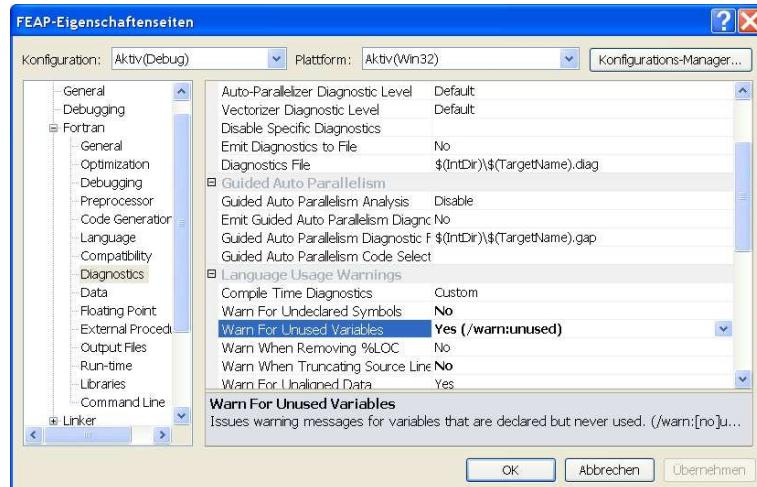
Compatibility>Enable F77 Run-time Compatibility: Yes

Compatibility>Use F77 Integer Constants: Yes

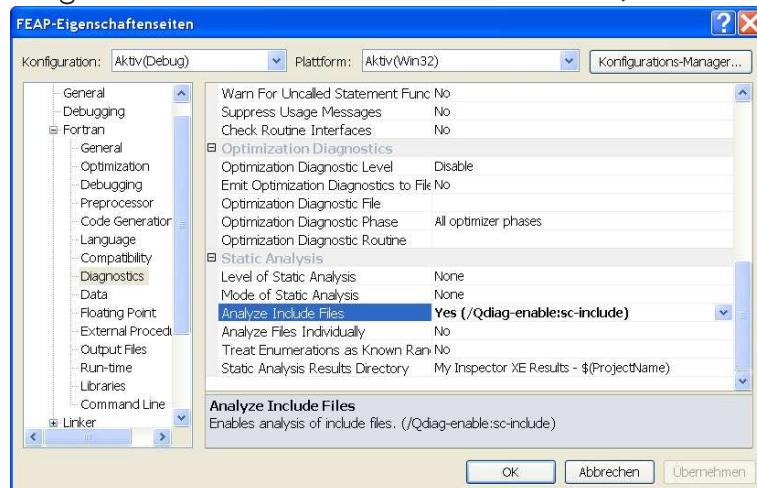
08-Diagnostics



Diagnostics>check routine interfaces: No

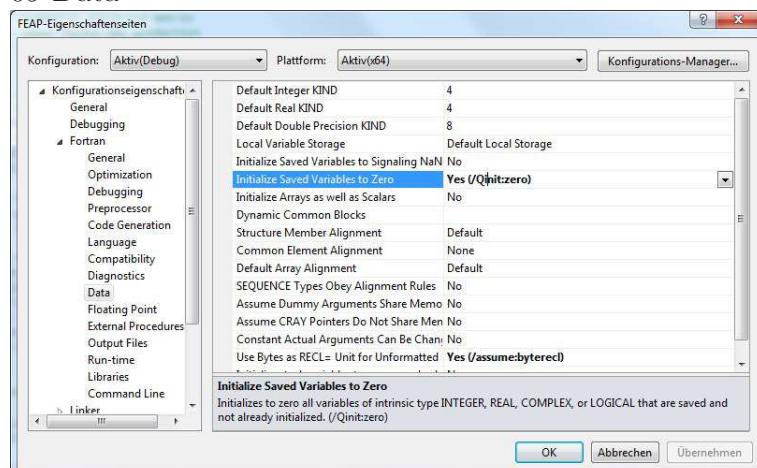


Diagnostics>Warn for unused variables; Yes (/warn:unused)



Diagnostics>Analyze include files: Yes (/Qdiag-enable:sc-include)

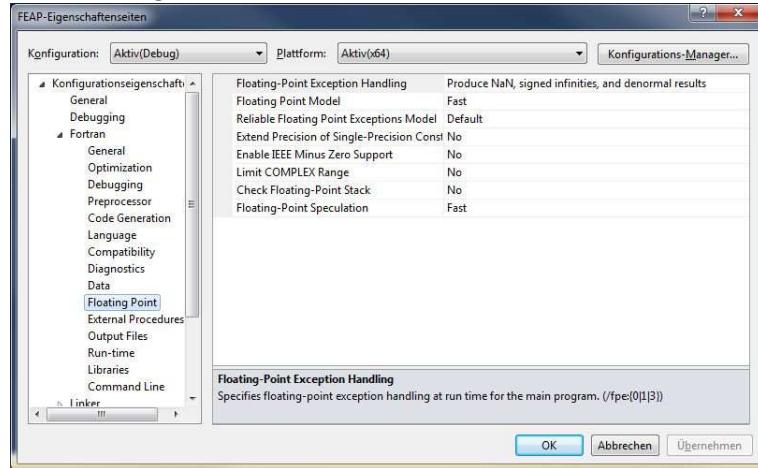
09-Data



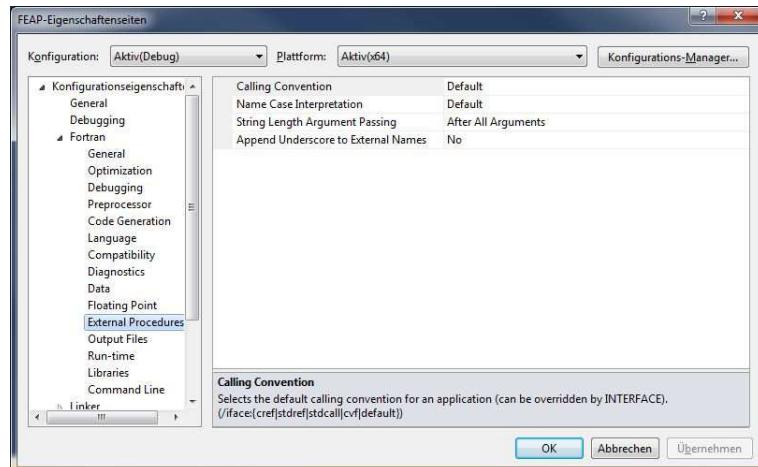
Data>Initialize Saved Variables to Zero: Yes

Data>Use Bytes as RECL: Yes (/assume:byterec1)

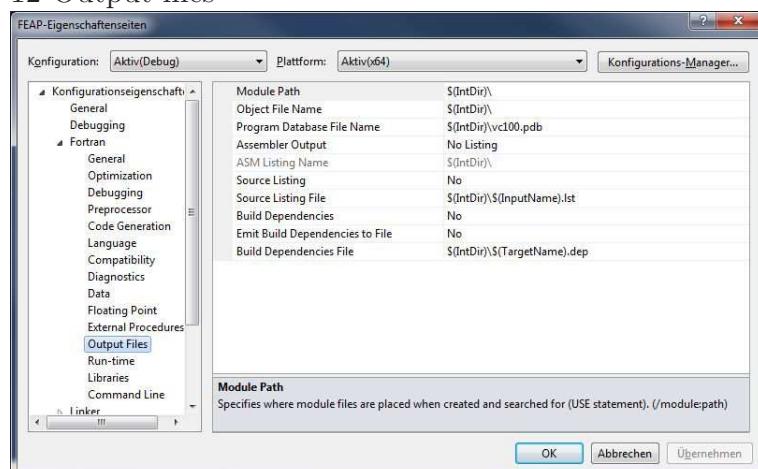
10-Floating Point



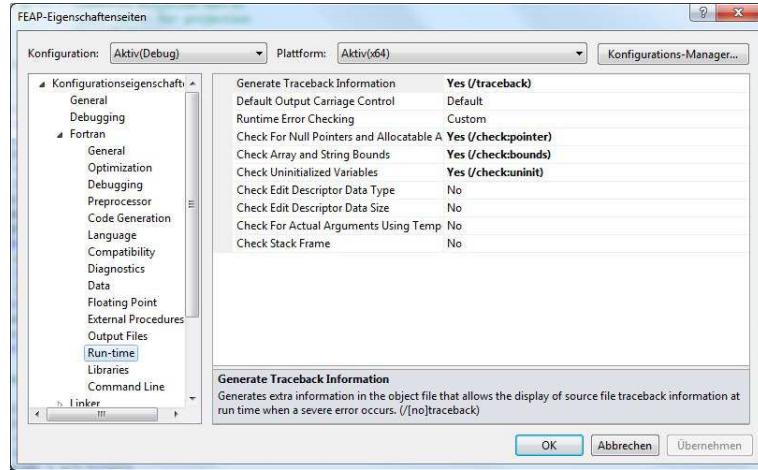
11-External Procedures



12-Output files

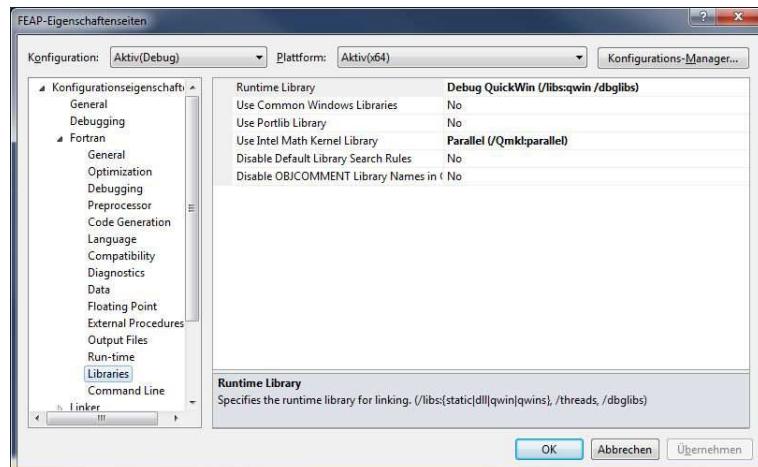


13-Run-time



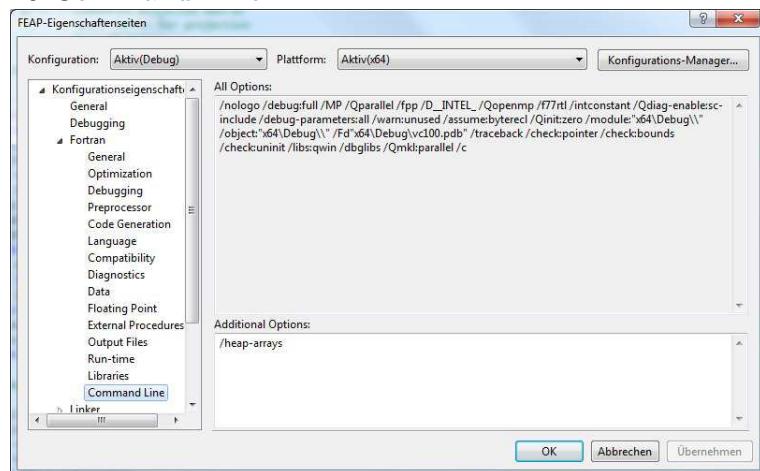
```
run-time>Generate Traceback Information: Yes  
run-time>Check for Null Pointers and Allocatable Array References: Yes  
run-time>Check Array and String Bounds: Yes  
run-time>Check Uninitialized Variables: Yes
```

14-Libraries



```
Runtime Library: Debug QuickWin (/libs:qwin /dbglibs)  
Use Intel Math Kernel Library: Parallel (Qmkl:parallel)
```

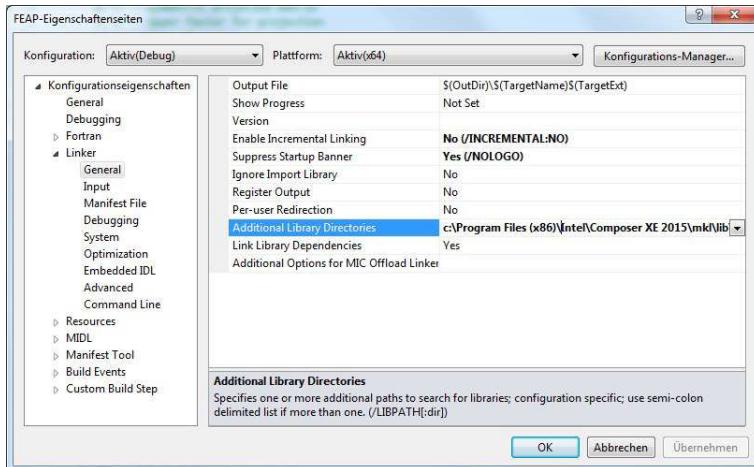
15-Command Line



Additional Options: /heap-arrays

3. Linker

01-General

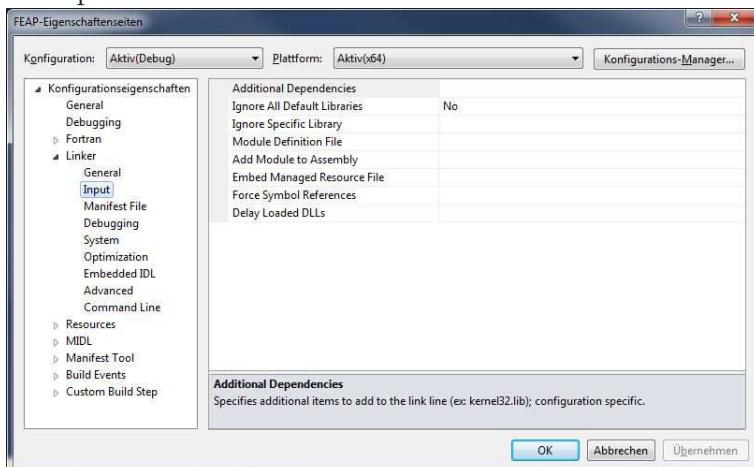


General>Additional Library Directories

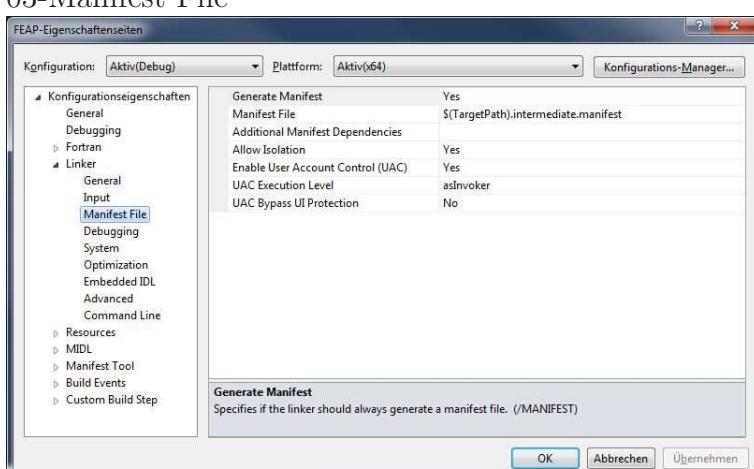
Angabe des Pfades zu den MKL-Bibliotheken:

WIN32: c:\Program Files (x86)\Intel\Composer XE 2015\mkl\lib\ia32\
WIN64: c:\Program Files (x86)\Intel\Composer XE 2015\mkl\lib\intel64\

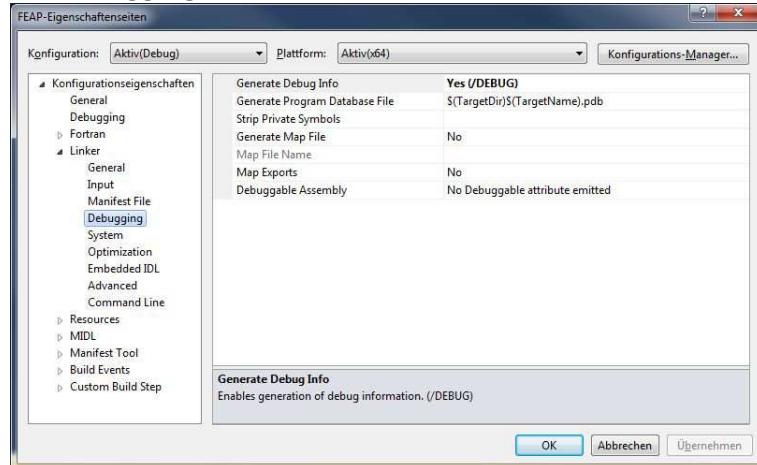
02-Input



03-Manifest File

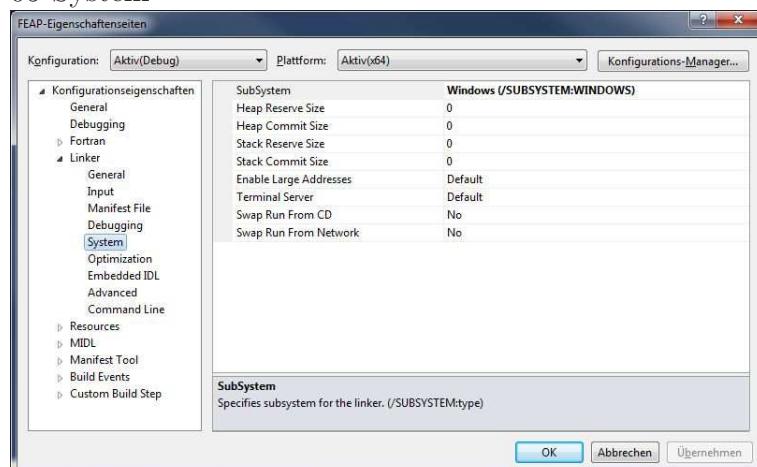


04-Debugging

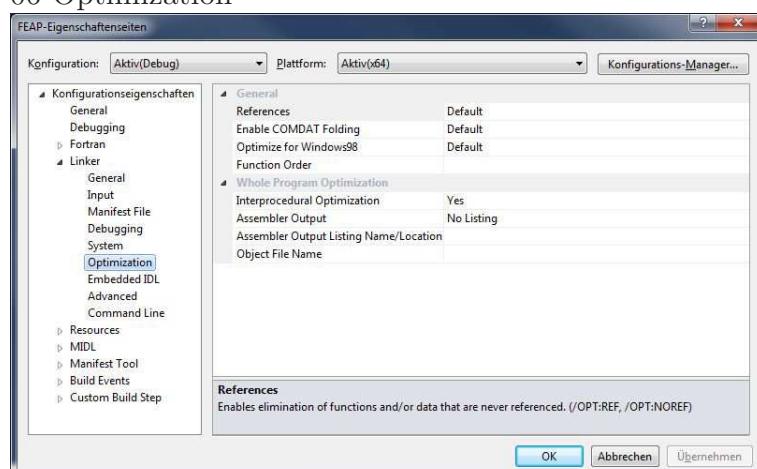


Generate Debug Info: Yes (/Debug)

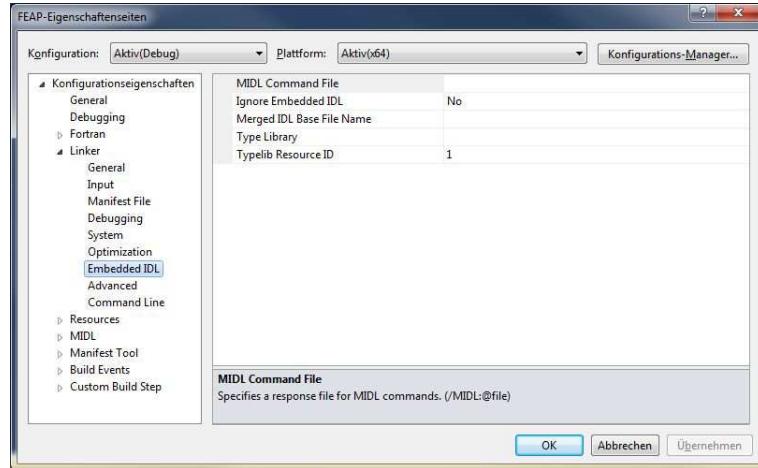
05-System



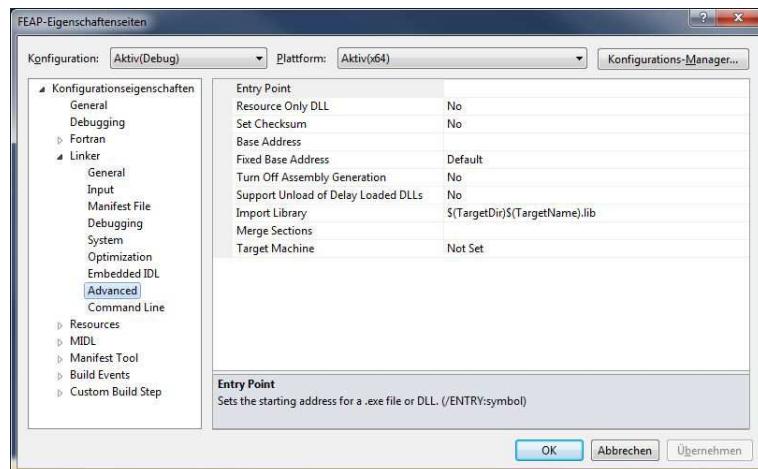
06-Optimization



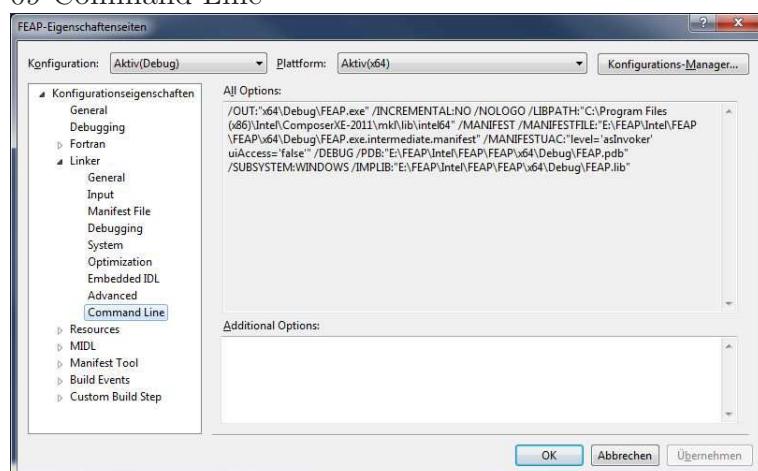
07-Embedded IDL



08-Advanced



09-Command Line

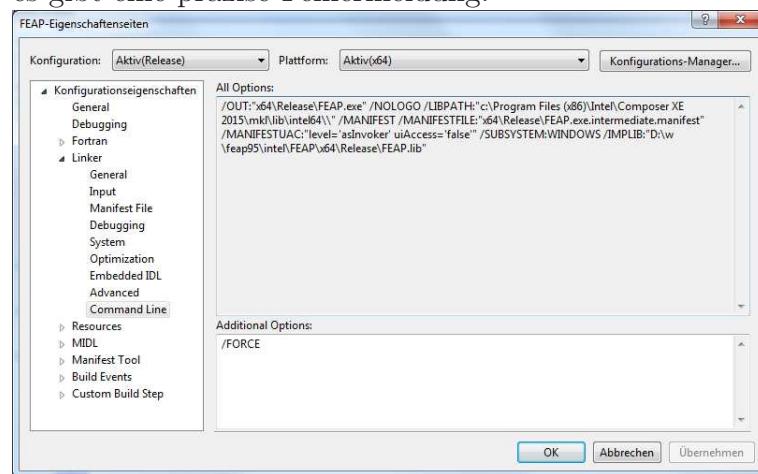


Additional Options:

- normalerweise nicht einfügen.
- optional ist es möglich den Linker zu zwingen auch eine unvollständige (fehlende Programmteile) EXE-Version zu erstellen. Die fehlenden Teile werden beim Linken dokumentiert. Die Vorgehensweise ist unkritisch, solange die fehlenden Teile nicht angesprochen werden.

Hierzu ist als Zusatzoption **/FORCE** zu setzen.

Treten Programmteile mehrfach auf (klassisches Beispiel: ELmtxx.for und gleichzeitig aktives Element Elmtxx in Elmlib.for) wird nicht gelinkt und es gibt eine präzise Fehlermeldung.



• FEAP-Projekteigenschaften (Compilier-Optionen-RELEASE)

Diese Version ist zum 'Schnellrechnen'. Alle Debug-Optionen entfernen, alle Optimierungs-Optionen auf höchste Stufe setzen, alle Parallel-Optionen beibehalten.

- **Make**

FEAP wird compiliert und gelinkt mit `project>build>build Feap.`

Dabei gibt es 3 Optionen. `build` übersetzt nur neue Dateien (das übliche **Make**) und `rebuild` übersetzt alle Dateien. Dies ist z.B. notwendig, wenn sich eine Include-Datei geändert hat. `clean` löscht alle Objects. `Build Solution` etc. macht das gleiche für eine `solution`. `solution` ist eine Summe von `builds`. Einzelne Programme können separat compiliert werden. Dazu im `Solution Explorer` die gewünschte Datei mit der rechten Maustaste anklicken und aus dem Menü `Compile` wählen.

- **Ergänzungen für FE²: Exe-Version für Hintergrundrechnung**

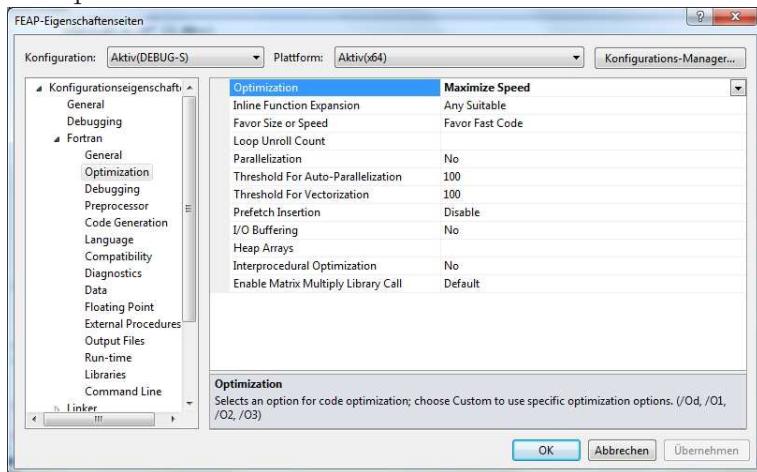
Für die **Mehrskalenrechnung** mit **FE²** ist eine zweite EXE-Version von FEAP zu erstellen. Diese wird mit dem Konfigurationsmanager angelegt. Name z.B. DEBUG-S. Es werden die Projekteigenschaften von DEBUG übernommen und danach abgeändert. Dabei sind die DEBUG-Einstellungen und die Parallelisierungseinstellungen zu entfernen sowie die Optimierungseinstellungen auf 'Optimal' einzustellen.

Die Änderungen sind wie folgt:

1. Projekteinstellungen

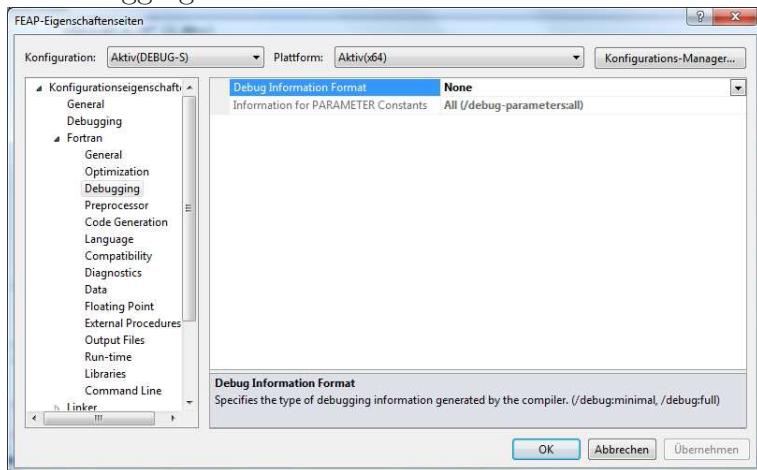
Fortran-Compiler

02-Optimization



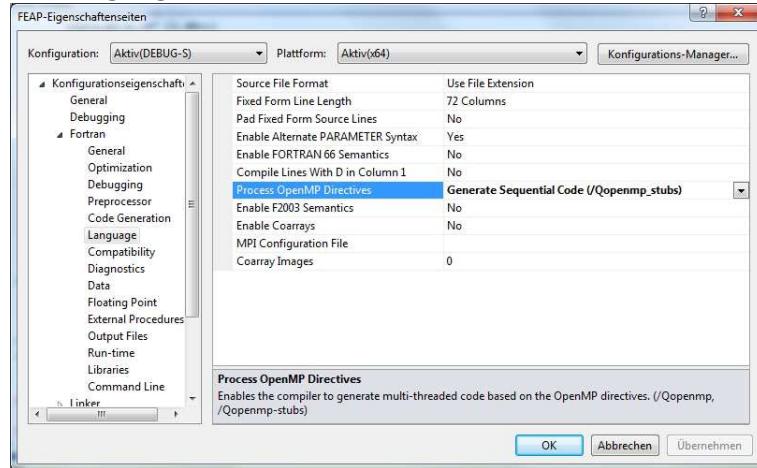
Optimization: Maximize Speed oder
Maximize Speed plus Higher Level Optimizations (/O3)
Parallelization: No

03-Debugging



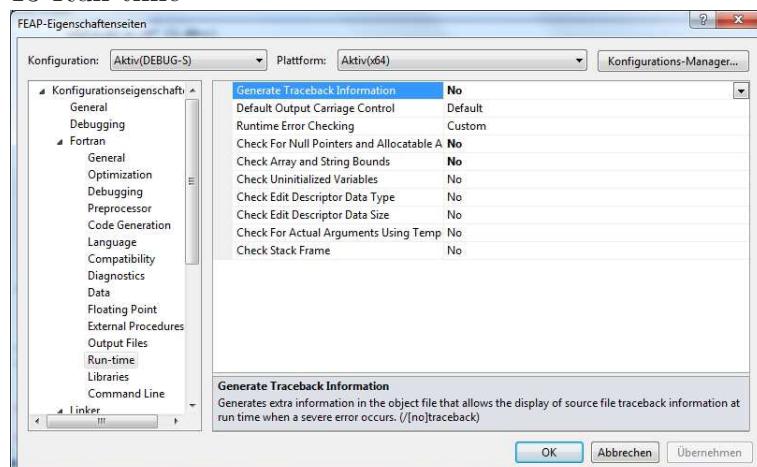
Debug Information Format: None

06-Language



Process OpenMP Directives: Generate Sequential Code (/Qopenmp_stubs)

13-Run-time



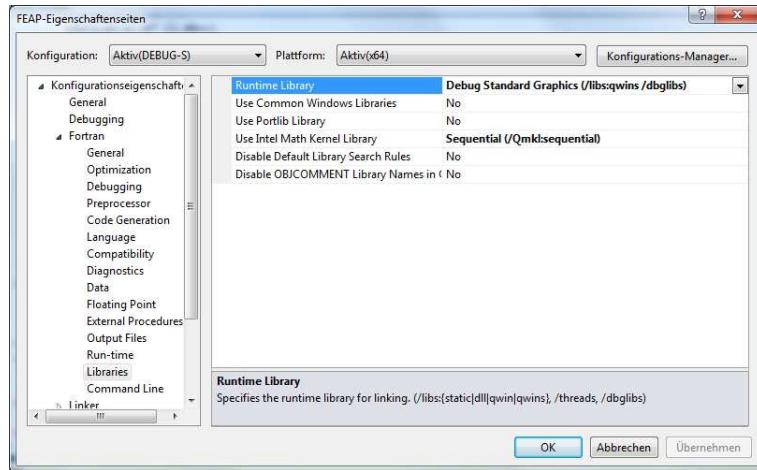
run-time>Generate Traceback Information: No

run-time>Check for Null Pointers and Allocatable Array References: No

run-time>Check Array and String Bounds: No

run-time>Check Uninitialized Variables: No

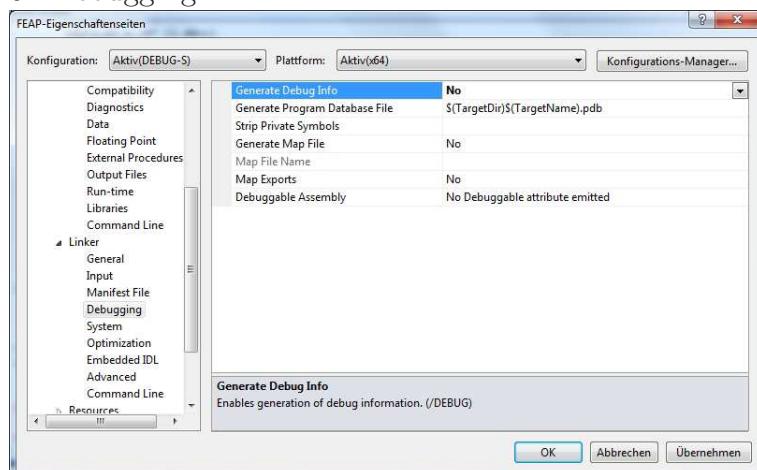
14-Libraries



Runtime Library: Debug Standard Graphics (/libs:qwins /dbglibs)
 Use Intel Math Kernel Library: Sequential (/Qmkl:sequential)

Linker

04-Debugging



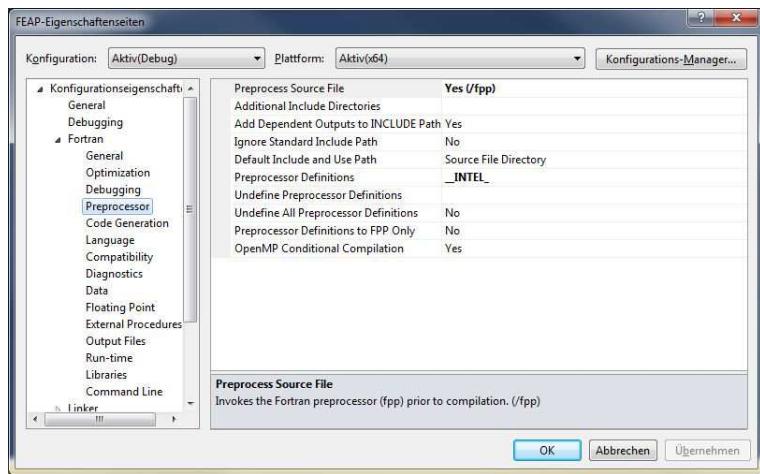
Generate Debug Info: No

2. In Feapwin.ini ist der Pfad zu dieser Version anzugeben:

```
feapint FEAP-INTEL
path\feap95\intel\feap\debug-s\feap.exe
```

3. Weitere Modifikation für ibin=2 (Shared Memory)

Falls nicht schon gemacht: in den Projekteinstellungen muss der Preprozessor aktiviert sein. Dazu: - Rechtsklick aufs Projekteigenschaften>Fortran>Preprocessor gehen



- Preprocess Source File: Yes (/fpp)

- Preprocessor Definitions: __INTEL__ (doppelter Unterstrich vorne, einzelner hinten)

Diesen Schritt für beide Build-Konfigurationen (mit und ohne Grafik) ausführen!

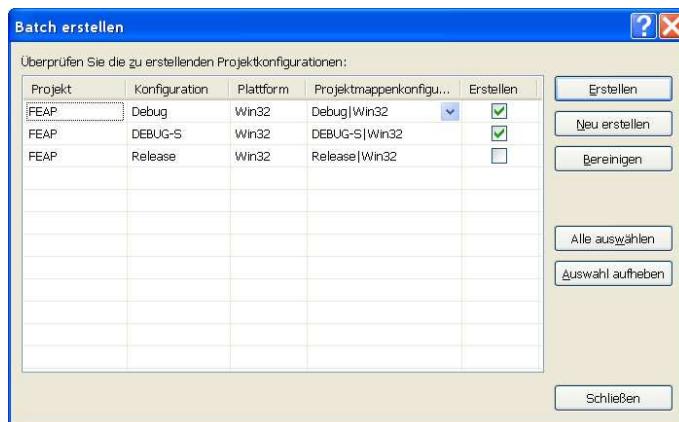
Intel rechnet dann standardmäßig mit Shared Memory bei FE². (ibin=2)

Anwendung derzeit in feap_micro.for, feap_shmem.for, mate3d08.for

4. Erstellen eines Batch-Files

Alle Änderungen(Compilieren, Linken) müssen unter DEBUG UND(!) DEBUG-S sowie RELEASE gemacht werden!!

Eine Automatisierung ist möglich, wenn man bei Projekteigenschaften das Erstellen eines Batch-Files wählt. Dort kann man anklicken, welche Versionen compiliert werden sollen.



9 Projekt FEAP-FTN in MS Visual Studio 2010 anlegen (IA32)

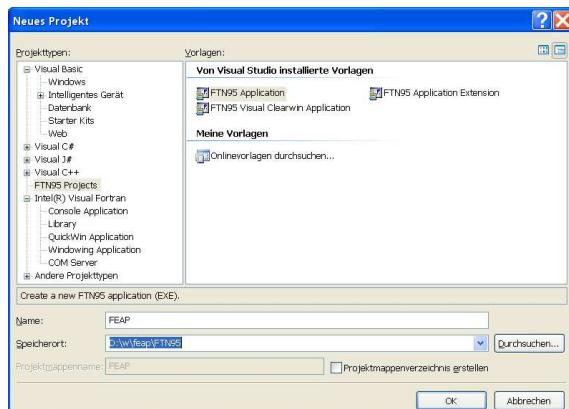
Hinweise: Alle Bilder und Menüs beziehen sich auf MS Visual Studio 2010 Professional und frühere Versionen. Das aktuelle Aussehen bzw. die Inhalte einzelner Menüs können ggf. hiervon abweichen!

- **FEAP-Quellen einbinden**

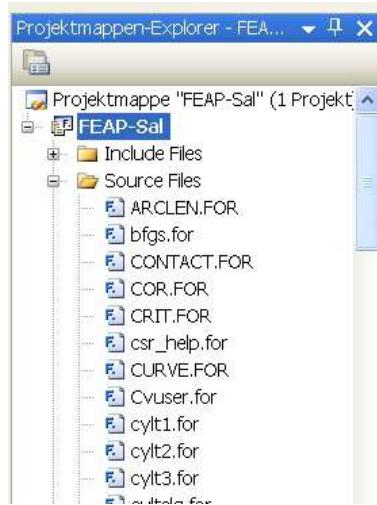
1. MS Visual Studio 2010 starten.
2. Projekt eröffnen: **File>New>project**

Type: **FTN95 Projects** und Template: **FTN95 Application**.

Der anzugebende Projektpfad ist nicht der Pfad zu den FEAP-Quellen, sondern der Pfad, wo später die Datei **FEAP.exe** sowie alle projektspezifischen Angaben liegen. Vorschlag: **FEAP\ftn95**.

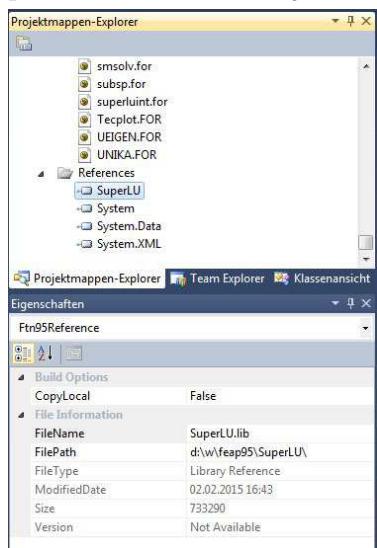


3. Projektmappenkonfiguration: **Debug**
4. Projektmappenplattform:
 - **WIN32**
Neue Plattformen werden mit dem Konfigurationsmanager (im Fenster Projektmappenplattformen) angelegt.
5. Fortran(source)-Quellen laden:
Im **Solution Explorer** (rechts) **FEAP Source Files** anklicken, dann mit **Project>Add Existing Items** alle für FTN95 erforderlichen Fortran-Quellen in das Projekt laden. Die **Include-Dateien *.h** sind nicht zu laden!
Ggf. im **Solution Explorer** die INTEL-Dateien mit der rechten Maustaste anklicken und aus dem Projekt löschen.
Das Ergebnis sieht so aus (Teilbeispiel!):



6. Bibliothek SuperLU einbinden

Im Solution Explorer (rechts) References anklicken, und die Bibliothek SuperLU.lib in das Projekt laden.



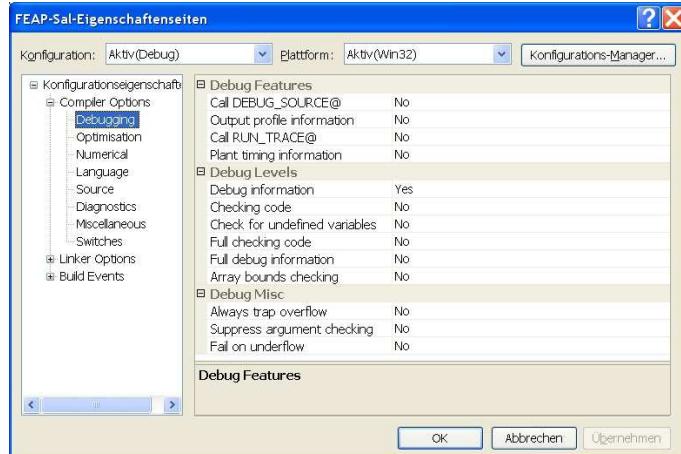
- **FEAP-Projekteigenschaften (Compilier-Optionen)**

1. Im Solution Explorer (rechts) FEAP anklicken, dann Project>properties anklicken.

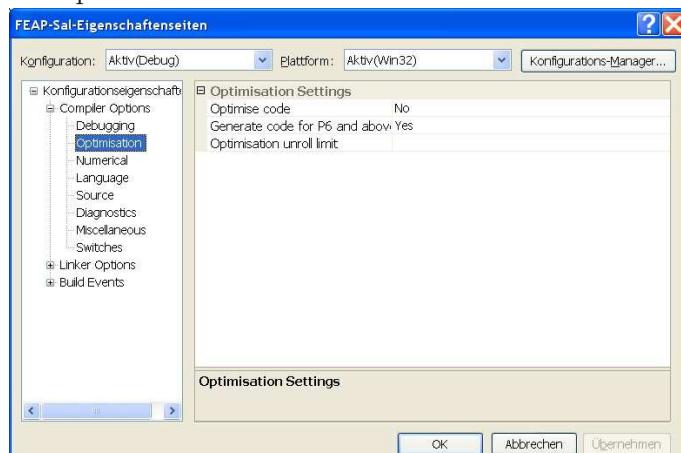
Danach sind die Compilier- und Linkeigenschaften in den Untermenüs einzustellen.

2. Fortran-Compiler

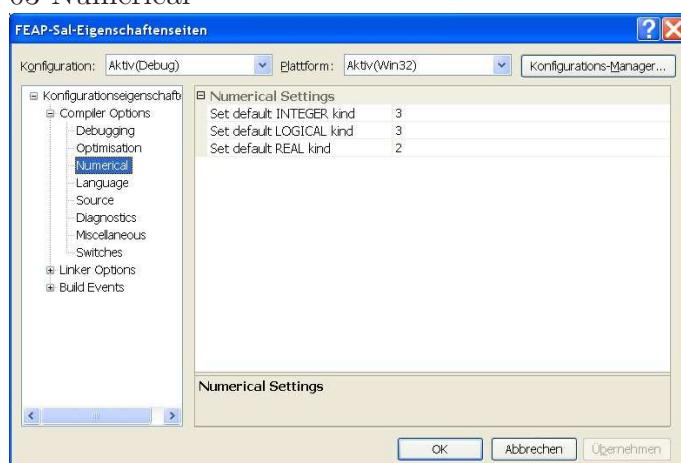
01-debugging



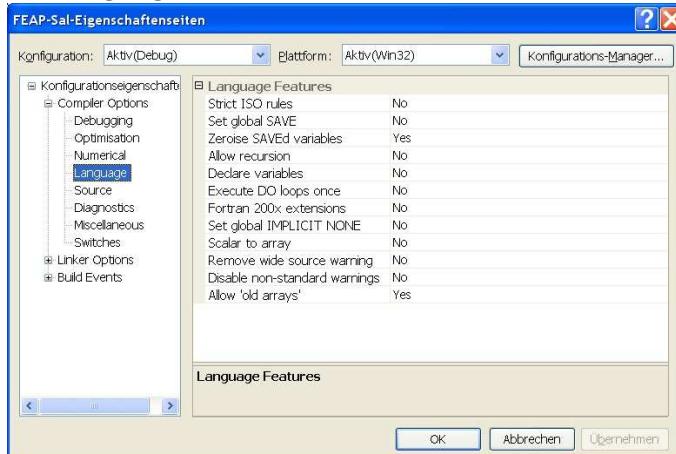
02-Optimisation



03-Numerical



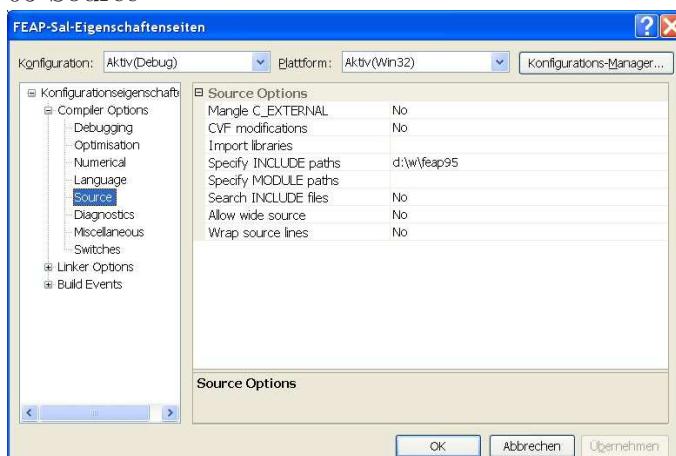
04-Language



Zeroise Saved Variables: Yes

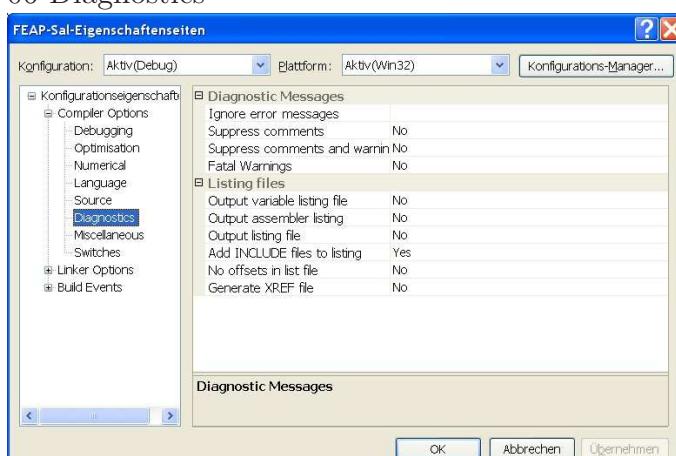
Allow old arrays: Yes

05-Source



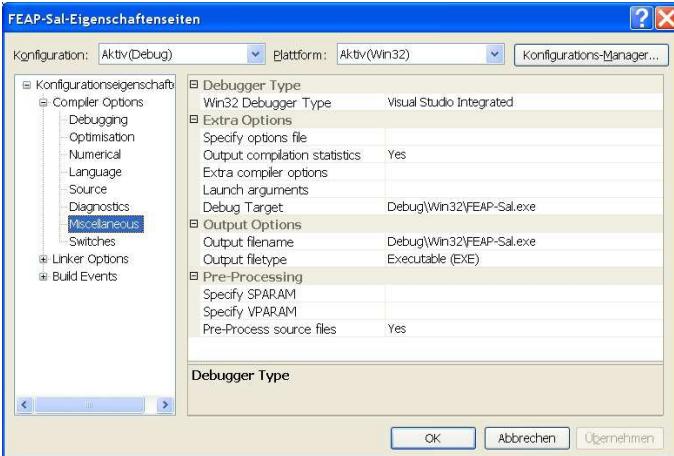
Specify INCLUDE paths: d:\w\feap95 (example for path to .h-files)

06-Diagnostics



Add INCLUDE files to listing: Yes

07-Miscellaneous-Änderung 1



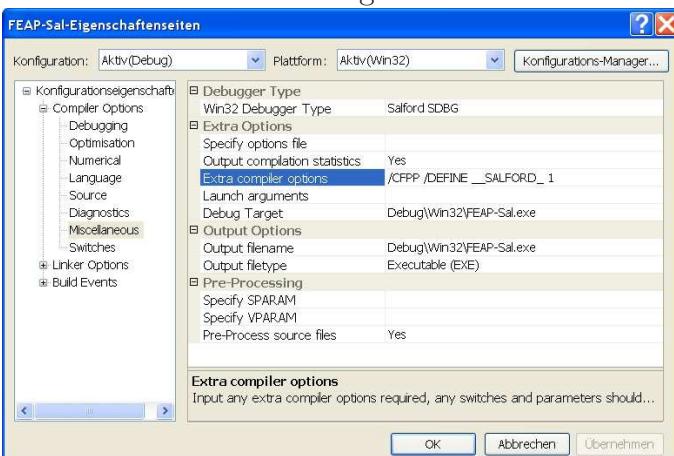
2 verschiedene Debugger können verwendet werden:

WIN32 Debugger Type: Visual Studio integrated oder Salford SDBG

Debug Target: Name is set automatically from project

Output filename: Name is set automatically from project

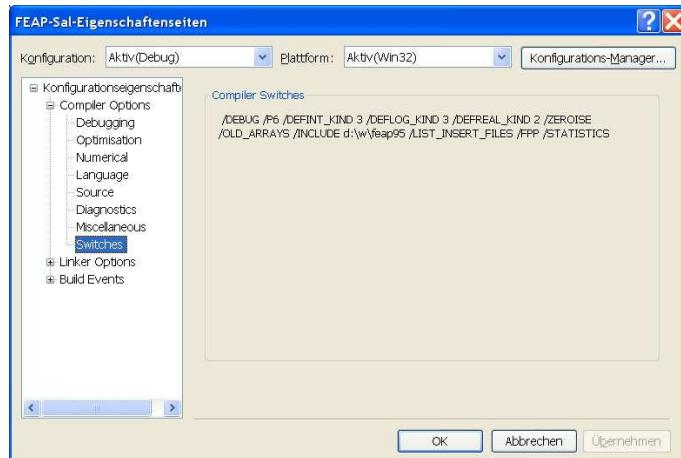
07-Miscellaneous-Änderung 2



- Extra compiler options: /CFPP/DEFINE __SALFORD_ 1

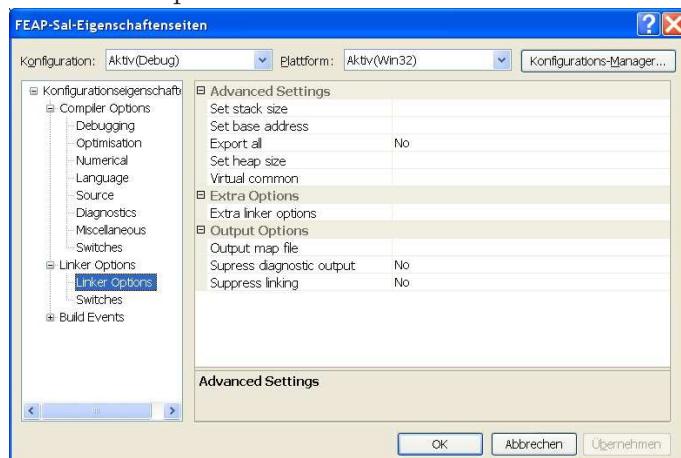
(für FEAPS5 und MATE3D08)

08-Switches



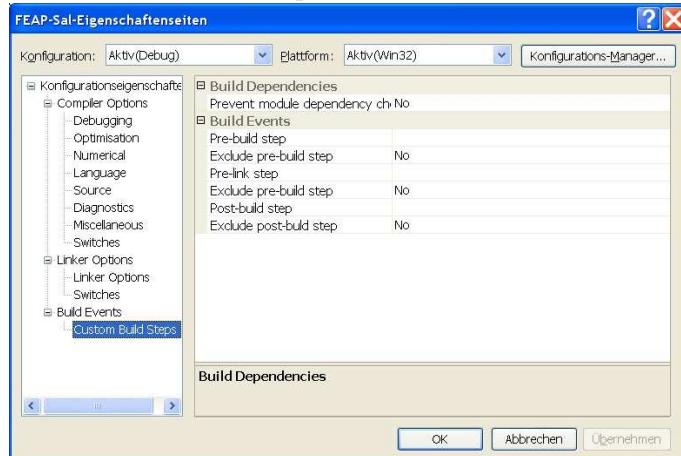
3. Linker

01-Linker Options



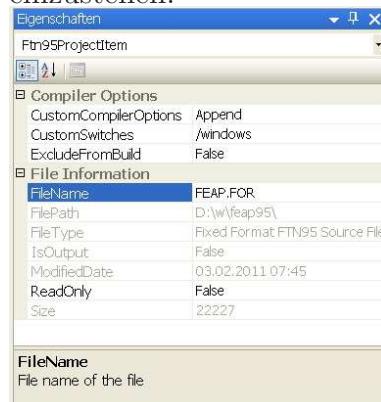
4. Build Events

01-Custom Build Steps



5. Compilier-Option /windows

Das Hauptprogramm(FEAP.for) und die Clearwinprogrammteile enthaltenen Dateien (plot_sal.for,plot_sal_cw.for) müssen mit der Option /windows compiliert werden! Dazu sind diese Dateien mit der rechten Maustaste im **Solution Explorer** (rechts) anzuklicken, dann ist unter **Project>properties** einzustellen.



CustomCompilerOptions: Append

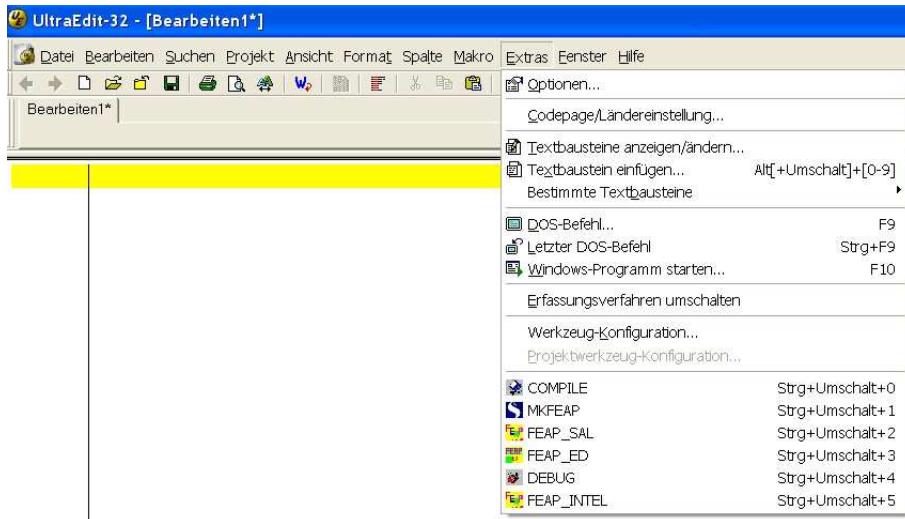
CustomSwitches: /windows

- Make FEAP wird compiliert und gelinkt mit project>build>build Feap.

Dabei gibt es 3 Optionen. build übersetzt nur neue Dateien (das übliche Make) und rebuild übersetzt alle Dateien. Dies ist z.B. notwendig, wenn sich eine Include-Datei geändert hat. clean löscht alle Objects. Build Solution etc. macht das gleiche für eine solution. solution ist eine Summe von builds. Einzelne Programme können separat compiliert werden. Dazu im **Solution Explorer** die gewünschte Datei mit der rechten Maustaste anklicken und aus dem Menü **Compile** wählen.

10 FEAP-FTN (IA32) mit Makefile in Ultredit

Alternativ zu MS Visual Studio kann FEAP auch unter Ultredit compiliert, gelinkt und gestartet werden. Dazu werden im Ultraedit unter Werkzeugkonfiguration verschiedene BAT-Files bzw. Prozeduren angelegt. Diese BAT-Files können natürlich auch separat (ohne Einbindung) gestartet werden.

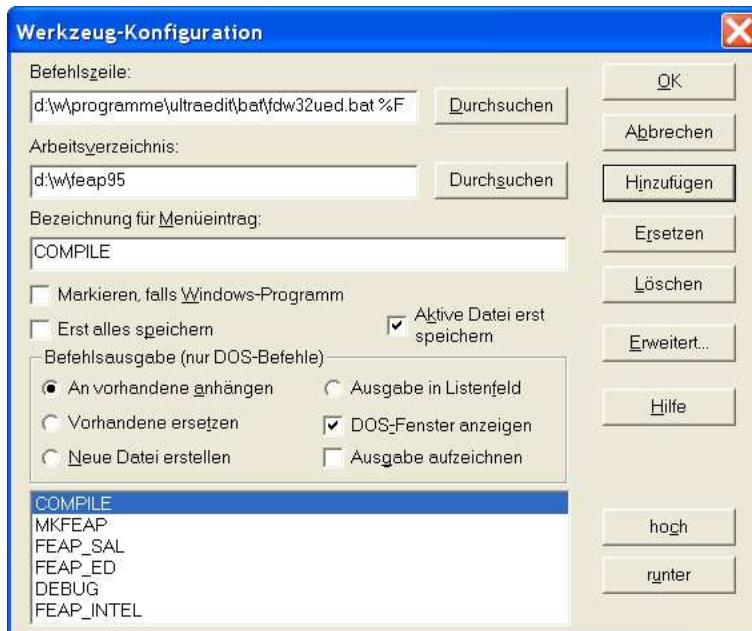


1. Compilieren einer Quelldatei

Der Compiliervorgang einschließlich Optionen ist in der Datei fdw32ued.bat abgelegt. Diese Datei kann an beliebiger Stelle abgelegt werden.

```
ftn95 %1 /intl /log1 /debug /CFPP /DEFINE __SALFORD_ 1 /windows>out
```

Das Ergebnis der Compilierung wird in die Datei OUT geschrieben. Compiliert wird die aktive Datei in Ultraedit.



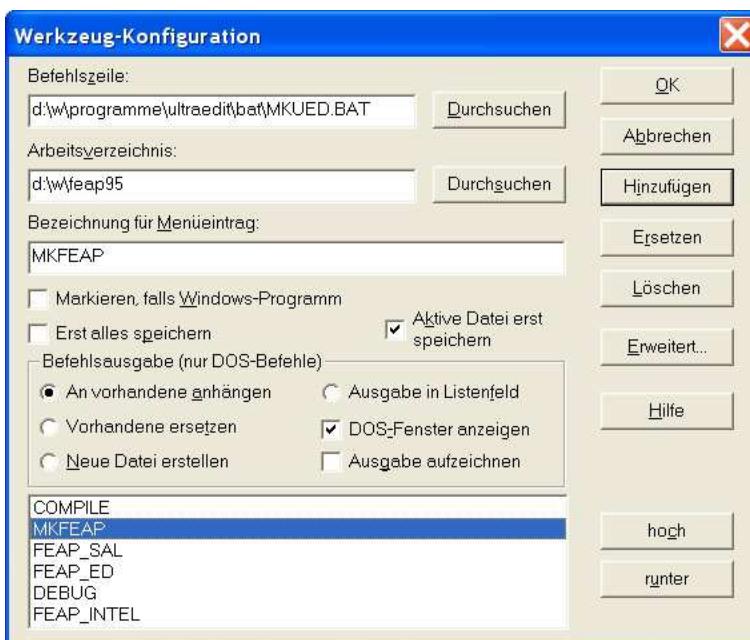
Ein zugehöriges ICON kann unter **erweitert** geladen werden.



2. Linken von FEAP

Der Linkvorgang ist in der Datei `mkued.bat` abgelegt.

```
mk32.exe  
pause
```



Dazu ist eine Datei `makefile` und eine Datei `link_f` anzulegen.

Datei `makefile`:

```
.SUFFIXES: .for .obj
```

```
ff      = /int1 /log1 /fullcheck /undef /CFPP /DEFINE __SALFORD_ 1
fd      = /int1 /log1 /debug /CFPP /DEFINE __SALFORD_ 1
fo      = /int1 /log1 /CFPP /DEFINE __SALFORD_ 1
foo     = /int1 /log1 /optimize /CFPP /DEFINE __SALFORD_ 1
ffw    = /int1 /log1 /fullcheck /undef /windows /CFPP /DEFINE __SALFORD_ 1
fdw    = /int1 /log1 /debug /windows /CFPP /DEFINE __SALFORD_ 1
fow    = /int1 /log1 /windows /CFPP /DEFINE __SALFORD_ 1
```

```
fcomp = ftn95
objs =
    module.obj \
    allocation_sal.obj \
    dalloc.obj \
    feap.obj \
    elmt01s.obj \
    elmt02s.obj \
    elmt03s.obj \
    elmt04s.obj \
    elmt05s.obj \
    elmt06s.obj \
    elmt07s.obj \
    elmt08s.obj \
    elmt09s.obj \
    elmt11s.obj \
    elmt12s.obj \
    elmt13s.obj \
    elmt14s.obj \
    elmt15s.obj \
    elmt16s.obj \
    elmt17s.obj \
    elmt18s.obj \
    elmt19s.obj \
    elmt20s.obj \
    elmt21s.obj \
    elmt22s.obj \
    elmt23s.obj \
    elmt24s.obj \
    elmt25s.obj \
    feapcfg.obj \
    feapico.obj \
    feaps1.obj \
    feaps2.obj \
    feaps3.obj \
    feaps4.obj \
    feaps5.obj \
    feaps6.obj \
    feap_micro.obj \
    fepost.obj \
    arclen.obj \
    bfgs.obj \
    contact.obj \
    cor.obj \
    crit.obj \
```

```
csr_help.obj \
curve.obj \
cvuser.obj \
cylt1.obj \
cylt2.obj \
cylt3.obj \
cyltalg.obj \
dplot.obj \
dynamics.obj \
eig1.obj \
elmlib.obj \
ext.obj \
gframe.obj \
help.obj \
hpgl.obj \
iniuser.obj \
lanczos.obj \
mat.obj \
matelib3d.obj \
mate3d01.obj \
mate3d02.obj \
mate3d03.obj \
mate3d04.obj \
mate3d05.obj \
mate3d06.obj \
mate3d07.obj \
mate3d08.obj \
mate3d09.obj \
mate3d10.obj \
mate3d11.obj \
mate3d12.obj \
mate3d14.obj \
mate3d15.obj \
pardisoint.obj \
pbcg.obj \
pgmres.obj \
pgmres2.obj \
pmacr.obj \
pmesh1.obj \
pplotf.obj \
plot.obj \
plot_sal.obj \
plot_sal_cw.obj \
pofeap.obj \
precond.obj \
```

```

remesh1.obj \
remesh2.obj \
remesh3.obj \
reme_nl1.obj \
reme_nl2.obj \
section.obj \
simplex_qld.obj \
smoo_control.obj \
smsolv.obj \
subsp.obj \
superluint.obj \
tecplot.obj \
ueigen.obj \
umfpack.obj \
umfint.obj \
unika.obj

feap.exe : $(objs)
           slink link_f

feap.obj      : feap.for
                 $(fcomp) feap.for $(fdw)

plot_sal.obj : plot_sal.for
                 $(fcomp) plot_sal.for $(fdw)

plot_sal_cw.obj : plot_sal_cw.for
                 $(fcomp) plot_sal_cw.for $(fdw)

smsolv.obj : smsolv.for
                 $(fcomp) smsolv.for $(fd)

.for.obj:
                 $(fcomp) $< $(fd)

```

Datei link_f:

```

lo module
lo feap
lo allocation_sal
lo dalloc
lo elmt01s
lo elmt02s
lo elmt03s

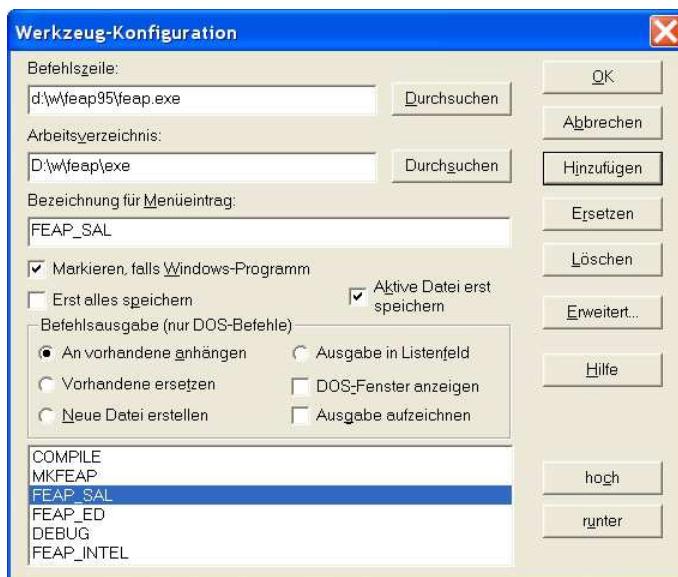
```

lo elmt04s
lo elmt05s
lo elmt06s
lo elmt07s
lo elmt08s
lo elmt09s
lo elmt11s
lo elmt12s
lo elmt13s
lo elmt14s
lo elmt15s
lo elmt16s
lo elmt17s
lo elmt18s
lo elmt19s
lo elmt20s
lo elmt21s
lo elmt22s
lo elmt23s
lo elmt24s
lo elmt25s
lo feapcfg
lo feapico
lo feaps1
lo feaps2
lo feaps3
lo feaps4
lo feaps5
lo feaps6
lo feap_micro
lo fepost
lo arclen
lo bfgs
lo contact
lo cor
lo crit
lo csr_help
lo curve
lo cvuser
lo cylt1
lo cylt2
lo cylt3
lo cyltalg
lo dplot
lo dynamics

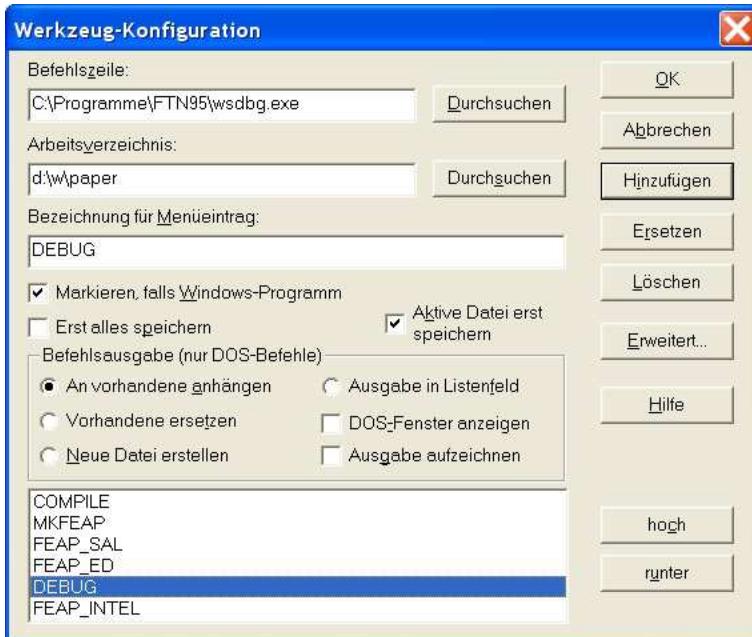
```
lo eig1
lo elmlib
lo ext
lo gframe
lo help
lo hpgl
lo iniuser
lo lanczos
lo mat
lo matelib3d
lo mate3d01
lo mate3d02
lo mate3d03
lo mate3d04
lo mate3d05
lo mate3d06
lo mate3d07
lo mate3d08
lo mate3d09
lo mate3d10
lo mate3d11
lo mate3d12
lo mate3d13
lo mate3d14
lo mate3d15
lo pardisoint
lo pbcg
lo pgmres
lo pgmres2
lo pmacr
lo pmesh1
lo pplotf
lo plot
lo plot_sal
lo plot_sal_cw
lo pofeap
lo precond
lo remesh1
lo remesh2
lo remesh3
lo reme_nl1
lo reme_nl2
lo section
lo simplex_qld
lo smoo_control
```

```
lo smsolv
lo subsp
lo superluint
lo tecplot
lo ueigen
lo umfpack
lo umfint
lo unika
lo superlu/superlu.lib
file feap
```

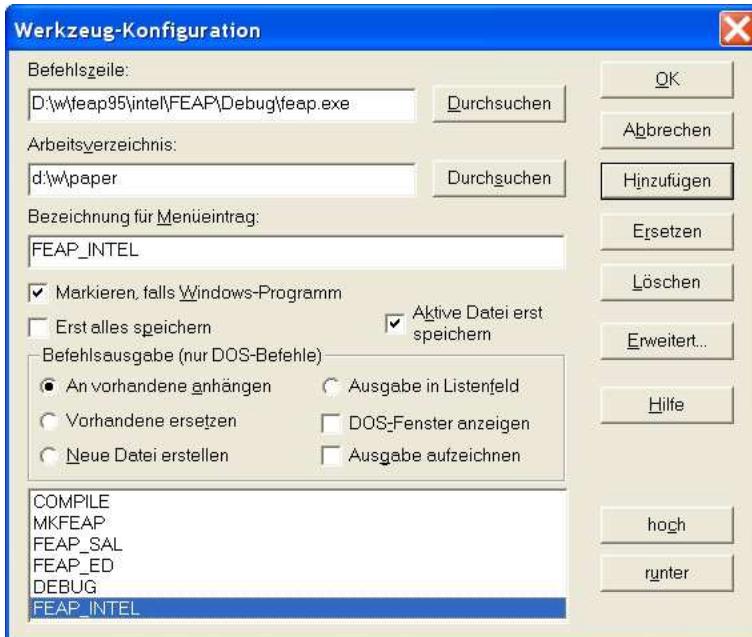
3. Starten von FEAP-SAL



4. Debuggen von FEAP-SAL



5. Starten von FEAP-INT



11 Anwendung der Debugger

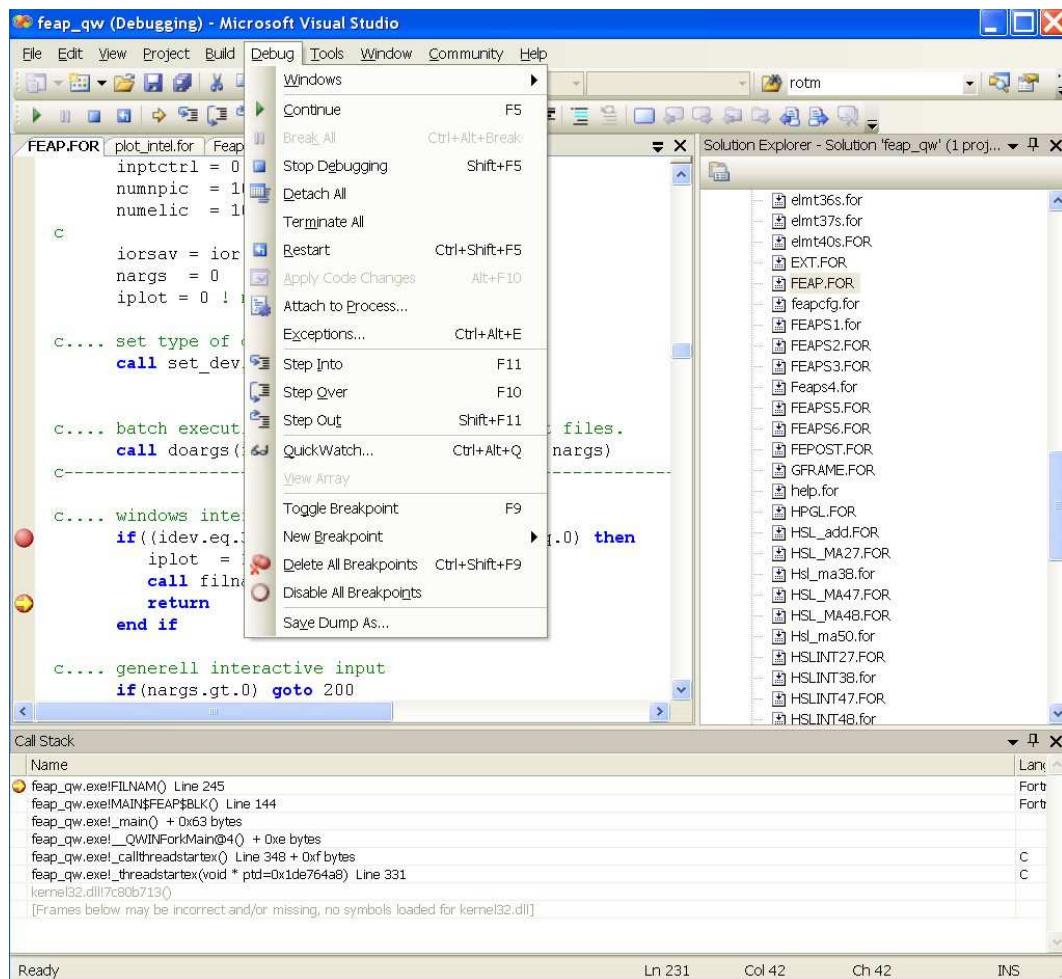
Der Debugger in MS-Visual Studio und der Salford-Debugger sind ähnlich aufgebaut.

• Visual Studio Debugger

Die Auswahlliste findet sich unter project>debug sowie alternativ mit Icons im Kopf von MS Visual Studio.

Die wesentlichen Befehle sind:

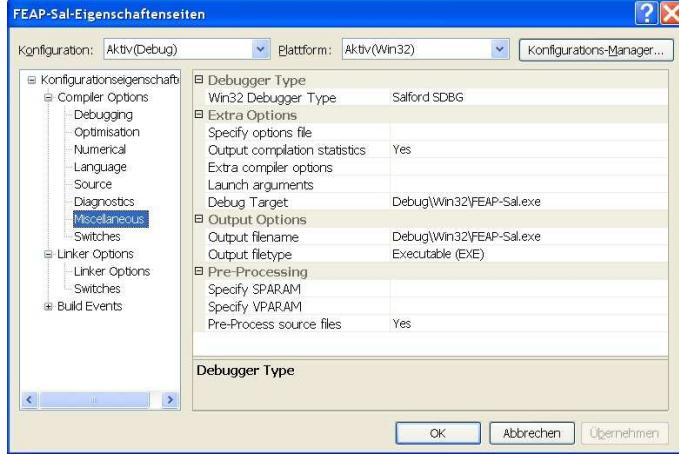
Aufgabe	Tastenkürzel
Start Debugger	F5
Stop Debugger	Shift+F5
Step Into	F11
Step Over	F10
Step Out	Shift+F11
Set Breakpoint	F9 oder l. Mausklick linker Rand
Liste der Breakpoints	Ctrl+Alt+B
Stack(Verlauf durch Programme)	Ctrl+Alt+C



• Salford Debugger

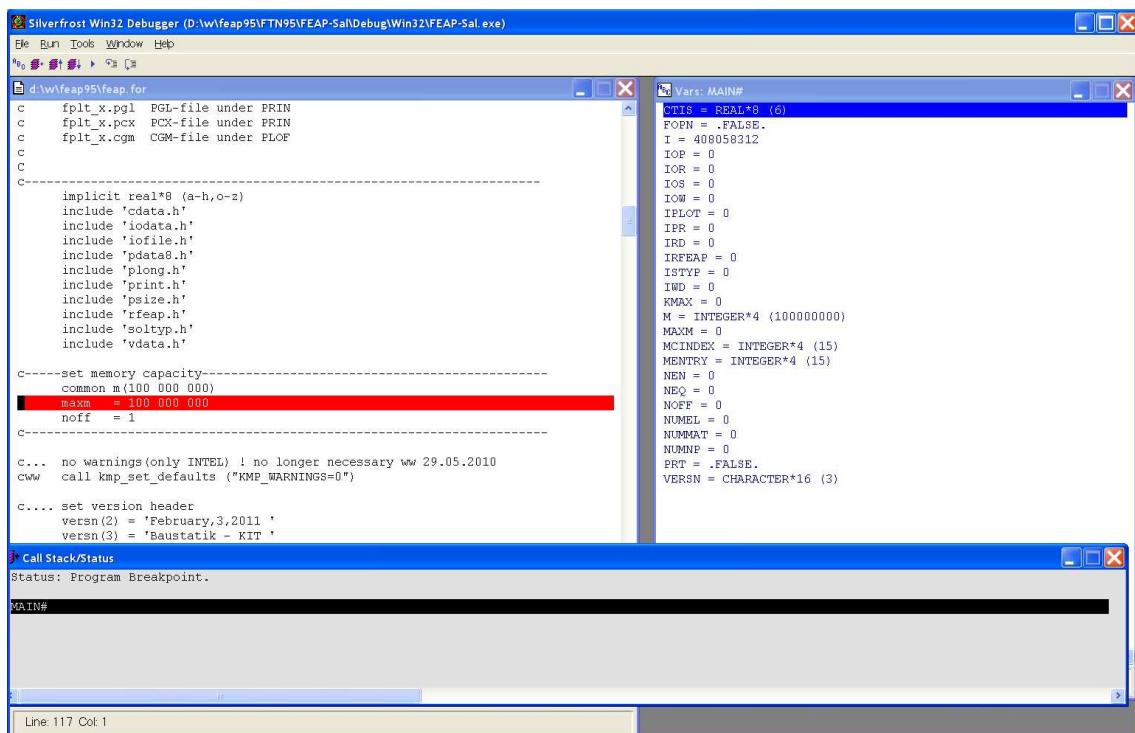
Der Salford Debugger kann entweder direkt unter
 c:\programme\ftn95\wsdbg.exe

angesprochen werden oder er kann über die Projekteigenschaften des Projektes FEAP-SAL eingestellt werden.



Die wesentlichen Befehle sind:

Aufgabe	Tastenkürzel
Continue	F6
Step	F7
Step Over	F8
Breakpoint	F2
Temporary Breakpoint	F3



12 Anwendung des Manuals

Es steht ein umfangreiches Manual zur Anwendung von FEAP zur Verfügung. Dabei gibt 2 Anwendungsstrategien:

- Unabhängig von FEAP oder auch aus FEAP heraus(z.B. Macro;Man) öffnet man das Manual einmal 'allgemein'. Danach kann man darin mit Hilfe der Bookmarks beliebig herumspringen.
- Man arbeitet in FEAP und kann an jeder Stelle z.B. mit HELP,TANG durch Sprung an die Stelle TANG direkte Hilfe zum verwendeten Macro bekommen. Das geht auf MESH,MACRO und PLOT-Ebene. Das Manual **muss nach jedem Aufruf geschlossen werden**. Nur dann wird beim nächsten Hilfe-Befehl in der Datei an die gewünschten Stelle gesprungen. Dies funktioniert auch in den weiteren Hilfsprogrammen.

13 Anwendung von Parallellösern

Nur in der Intel-Version kann mit dem parallelen Gleichungslöser PARDISO gerechnet werden.

- Definition im Input-File:

SOLV

12, isym, nproc

mit `isym=1` für symmetrische Probleme, `isym=2` für un-symmetrische Probleme und `nproc`: Zahl der zu verwendenden Prozessoren des Rechners (Default: alle Verfügbaren Kerne des Rechners).

`nproc` wirkt nur auf die mit OMP-parallel compilierten Programmteile. Der Löser verwendet unabhängig davon alle Prozessoren.

Der Parallelloser verwendet seine eigenes Speichermanagment.

Da es sich um einen direkten Löser handelt, wächst der Speicherplatzbedarf des Löser bei großen Problemen massiv. Am Ende eines jeden Ausgabefiles finden sich detaillierte Informationen zur Speicherbelegung.

14 offene Probleme und Infos

- FTN95

1. Die parallele Version des Solvers SUPERLU fehlt.

- INTEL

1. einzubauen: GMRES-MKL-Version mit ILUT und ILU0
2. einzubauen: MKL-Routinen für Matrizenmultiplikation etc.
3. Es muss das separate Programm Filnamw_i.exe zum Lesen der Verwendeten Dateien verwendet werden. Die direkte Einbindung -analog zu FTN - ist noch offen.

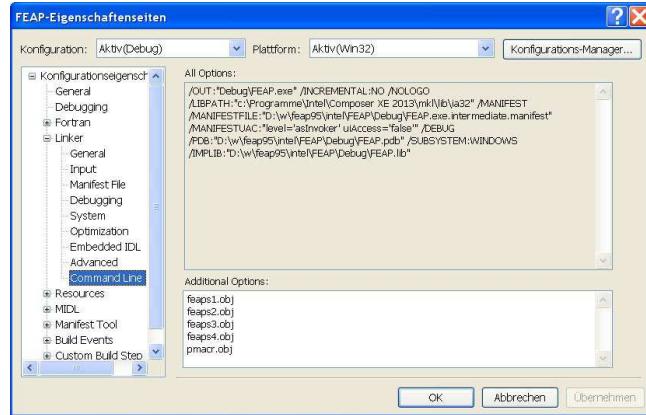
15 Erstellung von Studenten-Versionen

Für die Erstellung von studentischen Arbeitsversionen von FEAP werden dem Studierenden nur die für ihn notwendigen Quell-Dateien zur Verfügung gestellt.

- INTEL-in Visual Studio

Dazu ist wie folgt vorzugehen:

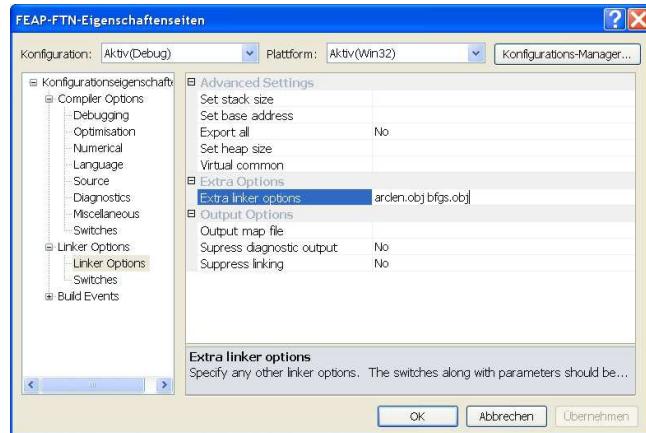
- Ein Projekt anlegen.
- Alle Quell-Dateien einfügen, compilieren und eine lauffähige Version erstellen.
- Dateien definieren, die nicht im Quell-Code benötigt werden.
- Im Projekt-Unterverzeichnis, typischerweise **Projektname>Debug** sind alle zugehörigen .obj-Dateien mit STRG X zu löschen.
- Diese sind dann mit STRG V in das Projekt-Verzeichnis(also da wo z.B. **FEAP.vfproj** liegt) zu verschieben.
- Alle zugehörigen Quell-Dateien aus dem **Projekt** entfernen.
- Alle zugehörigen Quell-Dateien aus dem **Quell-Verzeichnis** entfernen.
- **Die Datei module.for muss als Quell-Datei vorliegen, d.h. darf nicht aus dem Projekt und dem Verzeichnis entfernt werden.**
- Unter **Projekteigenschaften>Linker>CommandLine** die Namen der .obj-Dateien angeben. Mehrere Dateien können mittels eines Editors zeilenweise erstellt und dann hinein kopiert werden.



• SILVERFROST-in Visual Studio

Das Vorgehen ist analog zum INTEL-Projekt

- Unter Projekteigenschaften>Linker>ExtraLinkerOptions die Namen der .obj-Dateien angeben.
- Die Objekt-Dateien sind in das Projekt-Verzeichnis(also da wo z.B. FEAP.ftn95p liegt) zu verschieben.



• SILVERFROST-mit MAKE

- Makefile und Linkfile vollständig anlegen.
- Compilieren. Damit sind alle Object-Files angelegt.
- Dateien definieren, die nicht im Quell-Code benötigt werden. Diese Quellen löschen.
- Im Makefile diese Dateien ebenfalls löschen.

16 Verwendung

1. Die Quell-Programme von FEAP dürfen nur intern verwendet werden!
2. Eine Weitergabe an Dritte ist nicht erlaubt!
3. Für die Programme Microsoft Visual Studio, Intel Visual Fortran Studio, Silverfrost FTN95 und GotoAcro sind eigene Lizenzen zu verwenden!

17 FAQ - Installationsprobleme

1. FEAP-Lizenz nicht gefunden

- (a) Pfade bzw. Umgebungsvariable "FCFG=path\FEAP\add\" nicht richtig gesetzt
- (b) Lizenz tatsächlich abgelaufen, dann neue Lizenz holen

2. Programmabbruch oder fehlerhafte Rechnung

- (a) Pfade bzw. Variable
- (b) Umgebungsvariable "FCFG=path\FEAP\add\" nicht richtig gesetzt
- (c) Include-Dateien befinden sich in der Datei `Module.for`.
 - * Der FTN-95-Makefile erkennt Änderungen darin nicht. Daher alle .obj-Files löschen und vollständig neu compilieren.
 - * In Visual Studio/INTEL werden Änderungen erkannt. Im Zweifelsfall aber trotzdem 'FEAP neu erstellen' statt nur 'FEAP erstellen' anwenden.

3. Merkwürdige Fehlermeldungen unter Intel beim Compilieren

Wenn man beim Setzen der Projekteigenschaften nicht aufpasst, kann es passieren, dass bestimmte Eigenschaften nur für einzelne Files gesetzt werden. Dies ist dann der Fall, wenn man im rechten Fenster (Projektmappen-Explorer) versehentlich nur eine Datei markiert hat. Dies führt u.U. zu unverständlichen Fehlermeldungen. Markierung aller Dateien in diesem Fenster und setzen der Projekteigenschaften bereinigt das Problem in der Regel.

4. Manual-Zugriff springt immer an die selbe Stelle

Das Manual muss nach jedem Lesen geschlossen werden.

18 FEAP Shared Memory Dokumentation-Manual

Dominik Heller, Festkörpermechanik TU Darmstadt, heller@mechanik.tu-darmstadt.de

18.1 Einleitung

FEAP Shared Memory ist eine Erweiterung für FEAP. Ziel ist eine einfachere und effizientere Datenübertragung zwischen Makro- und Mikroprozessen von FEAP bei der Behandlung von Mehrskalenproblemen (FE²).

Bisher wurden die Problemdaten in Dateien auf der Festplatte geschrieben, was insbesondere bei großen Datenmengen hohe Lese- und Schreibzeiten verursacht. Bei Benutzung von FEAP Shared Memory werden Daten stattdessen im Hauptspeicher abgelegt und Zugriff sowohl von Makro- als auch Mikroprozessen ermöglicht (*shared memory*).

18.2 Voraussetzungen

Die Benutzung von FEAP Shared Memory ist nur unter Windows möglich. Zudem muss FEAP mit dem Intel Fortran Compiler erstellt werden – der Salford/Silverfrost FTN95 Compiler unterstützte benötigte Funktionen nicht.

18.3 Einrichtung

Siehe Installation der FEAP-Versionen!

18.4 Benutzung

FEAP Shared Memory bietet zwei separat voneinander benutzbare Module:

1. Spannungs- und Dehnungsdatenübertragung für alle FE²-Probleme (i.W. Felder *e*, *sig*, *cmat*)
2. History-/Restartfile Übertragung für plastische Probleme gemäß der FEAP-Routine *restrt* (**FEAPS5.FOR**)

Wichtig: Für beide Module muss FEAP in Windows Versionen ab Vista mit Administratorberechtigung gestartet werden (*Rechtsklick → Als Administrator ausführen*). Es ist zu empfehlen, sich eine dauerhafte Verknüpfung mit Administratorrechten zu erstellen (Verknüpfung erstellen, dann *Rechtsklick → Eigenschaften, Verknüpfung, Erweitert*, Haken setzen bei *Als Administrator ausführen*).

Zu 1:

Die Übertragung der Felder wird im Quellcode über den Parameter *ibin* gesteuert. Dieser ist zu setzen in der Routine *matm3d08* (**mate3d08.for**) sowie in den Routinen *fe2micro* und *epsq_ma* (beide **feap_micro.for**), die betreffenden Codestellen sehen jeweils wie folgt aus:

Hierbei ist

```
c...      transfer version
if(idev.eq.3) ibin=??? ! INTEL
```

- ibin=0: Datenübertragung über Dateien auf der Festplatte wie bisher
- **ibin=2**: Shared Memory Datenübertragung

Der Parameter *ibin* muss an allen 3 Codestellen sowie innerhalb Mikro- und Makro-FEAP gleich sein.

Wenn der Parameter geändert wird, müssen beide FEAP-Versionen(Mikro und Makro) neu kompiliert werden!

Zu 2:

Shared Memory als Ersatz für die Restartfiles wird innerhalb der Eingabefiles gesteuert, hierzu muss FEAP also nicht neu kompiliert werden.

Im Makroproblem ist im Material, nach den Pfaden zu Mikroeingabedatei und –batch Skript, der Parameter **2** (für Shared Memory) anzugeben, gefolgt von der Größe der Restartdateien in Bytes. Die Größenangabe kann eine Abschätzung nach oben sein. Es kann zu Abstürzen kommen, falls mehr Platz benötigt wird als angegeben wurde!

Beispiel Makro-Eingabedatei:

```
mate microproblem linear elastisch/elasto plastisch/Neo Hooke I/S = 60512 / 60498
1,21
8,0,1,
0
c:\fg\feap\exe\rve\4\irve4
c:\fg\feap\exe\rve\4\start4.bat
2,60498
```

In den zugehörigen Mikro-Eingabedateien muss innerhalb der *tang-* und *updh*-Kommandos an die Anweisungen *rest,,0* und *end,,0* der zusätzliche Parameter **,2** angehängt werden. *batc,micr* muss nicht angepasst werden.

Beispiel zugehörige Mikro-Eingabedatei:

<pre>batc,tang nopr prop,,1 rest,,0,2 epsq loop,,2 tang,,1 next sigq end,,0,2 1,1,0,10000,1 stop,tang</pre>	<pre>batc,updh nopr prop,,1 rest,,0,2 updh,,2 end,,0,2 1,1,0,10000,1 stop,updh</pre>
---	--

Innerhalb des Fortran-Codes kann auf den Parameter über die Variable *irtyp* zugegriffen werden, der Standardwert ist 0 (es werden Restart-Files auf der Festplatte verwendet).

Wechselt man häufiger zwischen Dateien und Shared Memory, so bietet es sich an die kompletten Problemdaten einmal zu kopieren und jeweils eine dateibasierte und eine Shared

Memory Version der Eingabedaten zu erzeugen.
Inkonsistente Eingabedaten sind eine häufige Fehlerquelle!

19 FEAP Shared Memory Dokumentation-Anhang

Dominik Heller, Festkörpermechanik TU Darmstadt, heller@mechanik.tu-darmstadt.de

19.1 Modifizierte Quelldateien

Für FEAP Shared Memory wurden folgende Quelldateien erstellt oder modifiziert:

- `feap_micro.for`
- `feap_shmem.for`
- `FEAPS5.FOR` (nur Modul 2, Shared Memory Restartdaten)
- `mate3d08.for`

Zudem wird eine neue Headerdatei benötigt:

- `shmname.h`

19.2 Known Issues - Probleme und Einschränkungen

Generell:

- **Benennung der Eingabedateien:**

Es kann zu Problemen kommen, falls die Namen der Eingabedateien (insb. Mikroprobleme) zu lang sind. Kurze Dateinamen wie `irve4_01` usw. sollten immer funktionieren. Leerzeichen in den Dateinamen sind unbedingt zu vermeiden.

FEAP Shared Memory erzeugt die Benennung der Speichersegmente aus den Inputdateinamen, z.B. der String `fres` in der `restrt`-Routine.

- **Maximale Elementzahl / Gaußpunktzahl:**

Derzeit ist durch die Benennung der Speichersegmente die maximale Anzahl an Elementen auf 10^6 begrenzt. Außerdem dürfen pro Element höchstens 999 Gaußpunkte auftreten. Bei Bedarf könnten diese Limits erhöht werden.

- **Parallele Elementschleife:**

In der neuesten FEAP Version (angepasste Transferchannels und Mikro-Inputfilebenennung anhand der Thread ID) funktioniert die parallele Elementschleife auch in FE²-Problemen.

Modul 2: (Restartdaten)

- Bei Benutzung einiger FEAP-Kommandos (bspw. `geom`) wurden im bisherigen Restart-Dateien-Ansatz `hrve...`-Dateien geschrieben, deren Inhalt jedoch offenbar nie benutzt. Das Schreiben dieser Dateien wird bei Benutzung von Shared Memory übersprungen (Routine `restrt`, `FEAPS5.FOR`):

```
#ifdef __INTEL__
    if(irtyp.eq.2) then
c...  mapping name and record length for shared memory restart file
        CALL feap_shmem_ParseRestartMappingName(fres,mapname)

        if (mapname(8:11).eq.'hrve') then
cDBG            WRITE (*,*) "Skipping hrve restart mapping"
cDBG            READ (*,*)
        RETURN
    end if
```

Es können ggf. Probleme auftreten, falls herkömmliche Restartdateien `hrve...` benannt werden.

19.3 Fehlermeldungen und Lösungsvorschläge

- Fehler bei Kompiliervorgang

Fehler beim Linken – fatal error LNK1104

```
Ausgabe
Ausgabe anzeigen von: Erstellen
C:\FEAP\ncgrf\FEAP\source\feap_shmem.for(1118): warning #6076: The data type of the actual s
mate3d08.for
plot_par.for
plot_intel_cw.for
Linking...
LINK : fatal error LNK1104: Datei "x64\Debug-mik\feap.exe" kann nicht geöffnet werden.

Build log written to "file:///C:/FEAP_ncgrf/FEAP/intel/feap_heb/x64/Debug-mik/BuildLog.htm"
FEAP_heb - 1 error(s), 77 warning(s)
===== Erstellen: 0 erfolgreich, Fehler bei 1, 1 aktuell, 0 Übersprungen =====
Codesdefinitionsfenster Aufrufbrowser Suchergebnisse: 1 Ausgabe
Fehler beim Linkvorgang
```

1. (Mikro-)FEAP-Prozess läuft noch. Im Taskmanager beenden (*Strg+Shift+Esc, Prozesse*, alle **feap.exe** Prozesse beenden).

Fehler beim Linken – error LNK2019: Verweis auf nicht aufgelöstes externes Symbol

```
Linking...
feap_micro.obj : error LNK2019: Verweis auf nicht aufgelöstes externes Symbol "FEAP_SHMEM_OPENNSMAPPING" in Funktion "FE2MICRO".
feap_micro.obj : error LNK2019: Verweis auf nicht aufgelöstes externes Symbol "FEAP_SHMEM_WRITECS" in Funktion "FE2MICRO".
feap_micro.obj : error LNK2019: Verweis auf nicht aufgelöstes externes Symbol "FEAP_SHMEM_CLOSEHANDLE" in Funktion "FE2MICRO".
FEAPSS.obj : error LNK2001: Nicht aufgelöstes externes Symbol "FEAP_SHMEM_CLOSEHANDLE".
mate3d08.obj : error LNK2001: Nicht aufgelöstes externes Symbol "FEAP_SHMEM_CLOSEHANDLE".
feap micro.obj : error LNK2019: Verweis auf nicht aufgelöstes externes Symbol "FEAP_SHMEM_OPENNSMAPPING" in Funktion "EPSQ_MA".
```

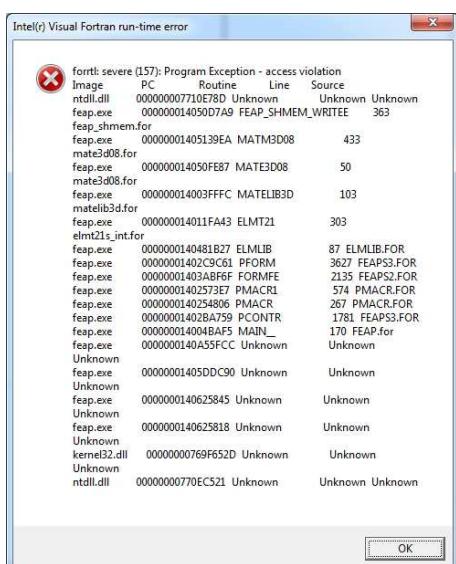
1. **feap_shmem.for** nicht ins Projekt eingebunden?
2. Präprozessoranweisung **_INTEL_** nicht gesetzt? Präprozessor aktiv? (**Yes /fpp**)

Compilerfehler – Cannot open include file

```
feap_micro.for
C:\FEAP\ncgrf\FEAP\source\feap_shmem.for(89): error #5102: Cannot open include file 'shmname.h'
C:\FEAP\ncgrf\FEAP\source\feap_shmem.for(89): error #5102: Cannot open include file 'shmname.h'
C:\FEAP\ncgrf\FEAP\source\feap_micro.for(892): warning #6049: This Holliechar or character constant is too long and cannot be used in the current numeric context. ['Global\FEAPmNNNTTIEEEEEE']
C:\FEAP\ncgrf\FEAP\source\feap_micro.for(892): error #6514: A substring must be of type CHARACTER. [SHMNAME]
C:\FEAP\ncgrf\FEAP\source\feap_micro.for(893): error #6503: The assignment operation or the binary expression operation is invalid for the data types of the two operands. [SLGP]
C:\FEAP\ncgrf\FEAP\source\feap_micro.for(897): error #6514: A substring must be of type CHARACTER. [SHMNAME]
C:\FEAP\ncgrf\FEAP\source\feap_micro.for(897): error #6503: The assignment operation or the binary expression operation is invalid for the data types of the two operands. [SLGP]
compilation aborted for C:\FEAP\ncgrf\FEAP\source\feap_micro.for (code 1)
```

1. Headerdatei **shmname.h** nicht vorhanden. Es können Folgefehler mit anderen Fehlermeldungen entstehen.

- Fehler zur Laufzeit



Absturz - Visual Fortran run-time error

1. FEAP im Administratormodus gestartet?
2. Präprozessordefinition **_INTEL_** gesetzt? (Makro und Mikro)
3. Größenangabe der Restartdateien im Makro-Eingabefile zu klein?
4. Parameter *ibin* und *irtyp* konsistent in Quell- und Eingabedateien?

Warning: Failed to create restart file mapping

1. FEAP im Administratormodus gestartet?
2. Größe der Restartdateien im Makro-Eingabefile (korrekt) angegeben?

```
FINITE ELEMENT ANALYSIS PROGRAM
(C) R.L. Taylor, University of California 2012

VERSION: WIN9x/NT/XP/WIN7
January,23,2012
Baustatik - KIT
STORAGE: 100000000

Macro>s,,5
Computing solution for time 0.100000E+01
Proportional load value is 0.100000E+01
EL,ISW,MAT      1       3       1
Warning: Failed to create restart file mapping. Please run with administrator privileges. (Global\rrve4_01.0001    Error code: 0
```

Warning: Could not open mapping

```
Warning: Could not open mapping 'Global\rrve4_01.0001'    Error code: 0
Warning: Could not open mapping 'Global\FEAPm001NSS000001'    Error code: 0
```

1. Parameter *ibin* und *irtyp* konsistent in Quell- und Eingabedateien?
2. FEAP im Administratormodus gestartet?

no input specified

```
no input specified
```

1. Genügend Mikro-Eingabedateien erzeugt? Bei paralleler Rechnung (OpenMP) wird pro Thread ein Eingabefile benötigt. Ggf. ist die `start*.bat` im Mikro-Verzeichnis anzupassen.

19.4 Fragen und Antworten

- Beim Kompilieren von `feap_shmem.for` treten haufenweise Compilerwarnungen auf??

Die Warnungen entstehen durch Datentypkonvertierungen zwischen Fortran und der Windows-C-API und können ignoriert werden.

```
C:\FEAP_nogfx\FEAP\source\feap_shmem.for(677): warning #6043: A dummy argument with an explicit INTENT(OUT) declaration is not given an explicit value. [R]
C:\FEAP_nogfx\FEAP\source\feap_shmem.for(677): warning #6043: A dummy argument with an explicit INTENT(OUT) declaration is not given an explicit value. [DETO]
C:\FEAP_nogfx\FEAP\source\feap_shmem.for(677): warning #6043: A dummy argument with an explicit INTENT(OUT) declaration is not given an explicit value. [DN]
C:\FEAP_nogfx\FEAP\source\feap_shmem.for(783): warning #6075: The data type of the actual argument does not match the definition. [LEN]
C:\FEAP_nogfx\FEAP\source\feap_shmem.for(783): warning #6043: A dummy argument with an explicit INTENT(OUT) declaration is not given an explicit value. [M]
C:\FEAP_nogfx\FEAP\source\feap_shmem.for(783): warning #6075: The data type of the actual argument does not match the definition. [LEN]
C:\FEAP_nogfx\FEAP\source\feap_shmem.for(813): warning #6075: The data type of the actual argument does not match the definition. [LEN]
C:\FEAP_nogfx\FEAP\source\feap_shmem.for(850): warning #6075: The data type of the actual argument does not match the definition. [LEN]
C:\FEAP_nogfx\FEAP\source\feap_shmem.for(819): warning #6043: A dummy argument with an explicit INTENT(OUT) declaration is not given an explicit value. [TTIM]
C:\FEAP_nogfx\FEAP\source\feap_shmem.for(819): warning #6043: A dummy argument with an explicit INTENT(OUT) declaration is not given an explicit value. [NBBM]
```

- Warum gibt es kein Shared Memory-FEAP für FTN95?

FTN95 unterstützt einige benötigte Windows-Plattformfunktionen nicht. Es gibt zwar eine compilereigene Shared Memory-Implementierung, diese ist jedoch verbuggt wenn verschiedene Prozesse derselben Executable (Mikro- & Makro-FEAP) auf ein Shared Memory-Segment zugreifen – laut Entwicklern ist diese Funktionsweise nicht beabsichtigt.

- Warum nicht etwas standardisiertes wie MPI zur Datenübertragung benutzen?

Bei großen Problemen treten schnell enorme Datenmengen auf, gerade bei der Benutzung von History-Daten. Eine schnelle Kommunikation zwischen Makro- und Mikroprozessen ist dann entscheidend, hierzu bietet sich ein Shared Memory-Ansatz an.

- **Warum die Präprozessoranweisungen?**

Es ist unkomplizierter und weniger fehlerträchtig als verschiedene Dateiversionen für die unterschiedlichen Compiler zu benötigen. Zudem wird der Präprozessor im Internen von FEAP Shared Memory sowieso benutzt und muss aktiviert werden.

- **Wie funktioniert FEAP Shared Memory im Wesentlichen?**

Statt in Dateien werden die Daten in benannten (vgl. Dateinamen) Speichersegmenten direkt im Hauptspeicher abgelegt und abgerufen. Auf diese Speichersegmente können alle Prozesse zugreifen, die deren Namen kennen. Die Segmente werden mithilfe der Windows-Plattformfunktionen *CreateFileMapping* und *MapViewOfFile* erzeugt und abgerufen.

- **Ich habe Fragen / Fehler gefunden / Verbesserungsvorschläge / Kritik / will mithelfen.**

Bitte an heller@mechanik.tu-darmstadt.de wenden.