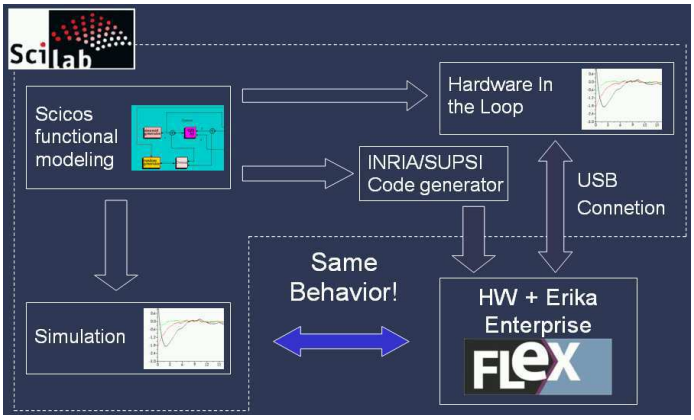
Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Prof. Dr.-Ing. Stefan Wiesemann

Professor für Technische Mechanik mechatronischer Systeme

Diplomarbeit

Untersuchung einer kostenlosen OpenSource-Simulationsumgebung für mechatronische Systeme



von

Hicham Azif

1.Prüfer : Prof.Dr.-Ing. Stefan Wiesemann

2.Prüfer : Dipl.-Ing. Helmut Rotter von IKS Engineering

Bearbeitunszeit : 10.11.2011-10.02.2012

Vorwort

Die vorliegende Diplomarbeit wurde an der Hochschule für Angewandte Wissenschaft (HAW) in Hamburg in dem Fachgebiet Maschinenbau erstellt. Die Betreuung erfolgte durch Herrn Pro. Dr. -Dipl.-Ing. Stefan Wiesemann, bei dem ich mich an dieser Stelle recht herzlich für die sehr gute Zusammenarbeit bedanken möchte. Des Weiteren möchte ich mich besonders bei meiner Familie, meiner Freundin und meinen Freunden für die großartige Unterstützung während des gesamten Studiums bedanken .

Erklärung

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbstständig und ohne unerlaubte Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich aus veröffentlichten oder unveröffentlichten Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Hamburg, den 10.02.2012

Hicham,Azif

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences
Department Maschinenbau und Produktion

A u f g a b e n s t e l l u n g

für die Diplomarbeit

von Herrn **Hicham Azif**

Matrikel-Nummer: **1851471**

Thema: **Untersuchung einer kostenlosen OpenSource-Simulationsumgebung für mechatronische Systeme**

Mechatronische Systeme beinhalten die vier Komponenten Sensorik, Aktorik, Mechanik und Informations- bzw. Elektrotechnik. Die Entwicklung solcher Systeme findet in der Regel in fünf Stufen statt:

1. SIL-Simulation (**S**oftware **I**n the **L**oop),
2. HIL-Simulation (**H**ardware **I**n the **L**oop),
3. RCP-Simulation (**R**apid **C**ontrol **P**rototyping),
4. Prototypenentwicklung und
5. Serienfertigung.

Während in der SIL-Phase noch alle vier der o.a. mechatronischen Komponenten simuliert werden, steigt der Anteil der realen Komponenten in den weiteren Entwicklungsstufen kontinuierlich an. Beim Prototypen werden erstmals alle mechatronischen Komponenten zu einem realen System verknüpft und die Funktionsfähigkeit untersucht. Die Entwicklung mechatronischer Systeme findet daher zunächst in einer entsprechenden Simulations-umgebung statt, die im Rahmen dieser Arbeit systematisch untersucht werden soll.

Schwerpunkte:

1. Installation der Simulationsumgebung (SciCos und ERIKA Enterprise)
2. Implementierung eines FLEX-Moduls
3. Überprüfung der Funktionsfähigkeit anhand einfacher mechatronischer Aufgaben
4. Entwicklung eines komplexen mechatronischen Systems
5. Erstellung eines kleinen Handbuchs zur Handhabung der Simulationsumgebung

Datum	1. Prüfer/in

Inhaltverzeichnis

Abbildungsverzeichnis

1.Einleitung

In dieser Diplomarbeit handelt es sich um die Untersuchung einer kostenlosen OpenSource-Simulationsumgebung für mechatronische Systeme.

Zu Beginn der Untersuchung sollte eine Installation der Simulationsumgebung von SciCos und ERIKA Enterprise vorgenommen werden. Um dies zu realisieren, muss eine ausführliche Installation von der neuen SciCos Version , Erika Enterprise und einer Implementierung von Flex-light Multibus Bord durchgeführt werden. Die Multibus Bord ist eine Daughter Bord, die auf der Oberseite des Flex-Light eingesteckt wird, basierend auf dem dspic-Mikrocontroller von der Firma Microchip .Es verfügt über 2 serienmäßig Slots, 2 CAN-Steckplätze, 1 SPI-Slot und 1 Ethernet-Steckplatz. Diese sind eine Reihe von kleinen Modulen, die auf der Oberseite des Demo Board / Multibus Boards eingesteckt werden . Das entwickelte Multifunktionsboard eröffnet die Möglichkeit, eine Vielzahl von Applikationen zu realisieren.

Scilab ist ein Kostenloses Open-Source Software-Paket für wissenschaftliche Berechnungen. Diese bietet Hunderte von Funktionen für allgemeine Zwecke und eine Vielzahl von speziellen Routinen für numerische Berechnungen und eine Funktionsbibliotheken - Toolboxes für Simulation, Optimierung, Systemanalyse, Regler Entwurf, Signalverarbeitung ...

Scicos ist eine Scilab Toolbox mit einem graphischen Block-Diagramm-Editor für die Erstellung und Simulation dynamischer Systeme und automatische C-Code-Generierung.

Scilab/Scicos ist eine Alternative zu kommerziellen Programmpaketen wie MATLAB/Simulink und MATRIXx/SystemBuild.

Erika Enterprise ist eine kostenlose Open-Source RTOS Umsetzung der ISO 17356-API (abgeleitet aus dem OSEK / VDX API). Diese bietet minimal einen 4.1 KB Flash-Echtzeit-Kernel (RTOS) für Single-und Multicore-Embedded-Systeme und unterstützt verschiedene Code-Generatoren,einschließlich Scicos-FLEX , ein Code-Generierung Toolbox auf ScicosLab, basierend EICASLab , der CAL-Sprache , und andere. (Erika Enterprise, 2012)

2 EINFÜHRUNG

2.1 Software

2.1.1 Scilab

Scilab ist ein numerisches Rechenprogramm, entwickelt für die Regelungstechnik und Signalverarbeitung. Als Alternative zu Matlab wurde es 1990 am Institut national de recherche en informatique et en automatique (INRIA) in Frankreich entwickelt und die Weiterentwicklung wird seit 2004 von einem Konsortium unter der Leitung des INRIA koordiniert. Dieses in C und Fortran programmierte Softwarepaket steht für Nutzung jeglicher Art kostenlos zur Verfügung. Ursprünglich wurde Scilab für Unix Workstations entwickelt, ist jedoch erhältlich. Die Funktionalität und die Syntax ist stark an das weit verbreitete numerische Rechenprogramm Matlab angelehnt. Zusätzlich gibt es integrierte Skriptkonverter von Matlab nach Scilab. Ähnlich zu Matlab, basiert Scilab auf Matrizenrechnung, wofür etliche implementierte Scilab Funktionen gibt, die das wissenschaftliche Rechnen mit Matrizen vereinfachen. Die Matrizen können aus verschiedenen Datentypen bestehen inklusive reellen und komplexen Zahlen, Funktionen, Polynomen, Zeichenketten und vielen mehr. Seine Stärke erhält Scilab erst durch die große Anzahl von Bibliotheken. Diese bestehen aus Funktionen, die in der Scilab eigenen Skriptsprache geschrieben sind. Viele der Bibliotheken sind auf die lineare Algebra, numerische Integration und Optimierung spezialisiert und enthalten auch Funktionen für die Analyse von nichtlinearen Funktionen und (impliziten) dynamischen Systemen. Durch die immer größere Anzahl von Programm-paketen, die zum Großteil auf der offiziellen Homepage http://www.scilab.org veröffentlicht werden, ergibt sich eine Vielzahl von Anwendungsgebieten. Es gibt zusätzlich einige graphische Möglichkeiten um 2D-, 3D- oder parametrische Plots und Animationen zu erzeugen. Diese können in verschiedensten Formaten (z.B. Postscripts, Gif, Postscript-Latex, Xfig) exportiert werden. (Kepler)

2.1.2 Scicos

Scicos (Scilab Connected Object Simulator) ist eins der wichtigsten Programmpakete aus Scilab. Dies enthält die Möglichkeit zur Modellbildung

und zur Simulation von dynamischen Systemen und entspricht dem Matlab Paket Simulink. Mittels verschiedener Blöcke, die entweder aus Bibliotheken gewählt oder selbst implementiert werden, kann man zeitkontinuierliche oder zeitdiskrete Systeme simulieren. So ist es möglich komplexe Systeme mittels Knopfdruck auf Plausibilität zu prüfen. Der große Vorteil liegt ähnlich zu Simulink im modularen Aufbau von Scicosdateien. Funktionen, Systemen, Subsystemen und viele mehr werden in Scicos durch I/O Blöcke charakterisiert. Damit können Parameter des Systems meist mit einem Klick verändert werden. Ein Großteil der Standardblöcke, die häufig gebraucht werden, sind bereits in Scicos enthalten. Falls diese nicht ausreichen, können benutzerdefinierte Blöcke implementiert werden. Für diese selbst erstellten Blöcke gibt es verschiedene Möglichkeiten der Implementierung. Einerseits kann man aus schon bestehenden Standardblöcken ein Subsystem zusammenfügen und dieses Subsystem als Block speichern. Des Weiteren gibt es die Möglichkeit eine eigene Funktion mit Hilfe der Flag-Schreibweise zu verfassen. Dies kann u.a. in der Scilab Skriptsprache, in C oder in Fortran erfolgen. Für die Kompilierung der benutzerdefinierten Blöcke ist ein C Compiler erforderlich. (Kepler)

Mit Scicos können Sie:

- Grafisch modellieren, kompilieren und dynamische Systeme simulieren.
- Kontinuierliche und zeitdiskrete Verhaltensweisen im selben Modell kombinieren.
- Modell-Elemente-Paletten von Standard-Bausteinen auswählen .
- Neue Blöcke in der C, Fortran, oder Scilab Sprache programmieren.
- Run-Simulationen im Batch-Modus der Scilab Umgebung
- Generierung von C-Code aus dem Scicos Modell mit einem *Code-Generator.*
- Run-Simulationen in Echtzeit und reale Geräte mit Scicos-HIL. (INRIA, Paris-Rocquencourt center, 2012)

2.1.2.1 Scicos Formalismen

Scicos ist ein Programm für die Entwicklung von reagierenden Systemen. Scicos Modelle wurden mit Hilfe eines Block-Diagramm-Editors entwickelt, dem eine Sprache hinterlegt ist, die alle gut definierten Formalismen bereitstellt.

Diese Formalismen sind sehr einfach, da sie sich ausschließlich mit dem reagierenden Teil der Entwicklung befassen. Diese stellen nicht eine komplette Programmiersprache bereit. Die Blöcke werden in Scilab als Atome angesehen. Der Scicossemilator betrachtet sie fast als schwarze Boxen. Es weiß einige ihrer Eigenschaften, aber nicht den dahinterliegenden Code. Der Code erkennt das Verhalten eines Blocks (die sogenannte Simulations-funktion), welches als ein C geschrieben werden kann, Fortran oder Scilab. In Scicos Formalismen wird die Ausführung von Simulationsfunktionen als unverzögert angesehen, damit kann Scicos als eine synchrone Sprache oder etwas mehr spezifisch als eine Verlängerung eines bearbeitenden weiterführenden Zeitsystems angesehen werden. (Nikoukhah)

2.1.4 Scicos und Scilab

Scicos ist eine **Scilab** Toolbox und läuft in der Scilab-Umgebung. Der Zugang zu Scilab sowie dessen Funktionen sind bei der Gestaltung von Simulationsmodellen von großer Bedeutung.

1. **Scicos** Benutzer müssen oft Scilab Funktionen verwenden wie z.B. solche die für das Entwickeln von Filtern für die Signalverarbeitung oder für die Kontrollentwicklung in der Konstruktion von Simulationsmodellen bestimmt sind.
2. Die Scilab Programmiersprache kann im generellen verwendet werden oder dafür einen Prozess von mehreren Simulationen zu bündeln. Modelle, entwickelt von Scicos können als Funktion in Scilad verwendet werden. Scilab's graphische Anlagen können für eine schnelle Nachbearbeitung von Simulationsergebnissen verwendet werden. (Nikoukhah)

2.1.3 MPLAB IDE

MPLAB Integrated Devlopment Environment (IDE) ist ein freies ,integriertes Toolset für die Entwicklung von Embedded-Anwendungen wie Microchip PIC ®

und dsPIC ®-Mikrocontroller. MPLAB IDE läuft als 32-Bit-Anwendung auf Basis von MS Windows ®, ist einfach zu bedienen und enthält eine Reihe von freien Software -Komponenten für die schnelle Anwendungsentwicklung und super-aufgeladen Debugging.
MPLAB IDE dient auch als eine einzige, einheitliche grafische Benutzeroberfläche für zusätzliche Mikrochips und Drittanbietern Soft und Hardware-Entwicklungstools. Der Wechsel zwischen Werkzeugen ist ein Kinderspiel, und ein Upgrade von dem kostenlosen Software-Simulator für Hardware-Debug-und Programmier-Tools ist in einem Flash getan, weil MPLAB IDE verfügt über die gleiche Benutzeroberfläche für alle Werkzeuge. (Microchip)

2.3 Software-in-the-loop

Die Überführung der Steueralgorithmen in eine Programmiersprache, die von der Zielplattform unterstützt wird (z.B. „C“), kann manuell oder automatisch erfolgen. Wenn der Quellcode für die Zielplattform vorliegt, besteht bei einigen Simulatoren die Möglichkeit, diesen in spezielle Blöcke im Systemmodell einzubinden. Dieses Verfahren ist als „Software-in-the-loop“ (SIL) bekannt (s. a. Abbildung 1).

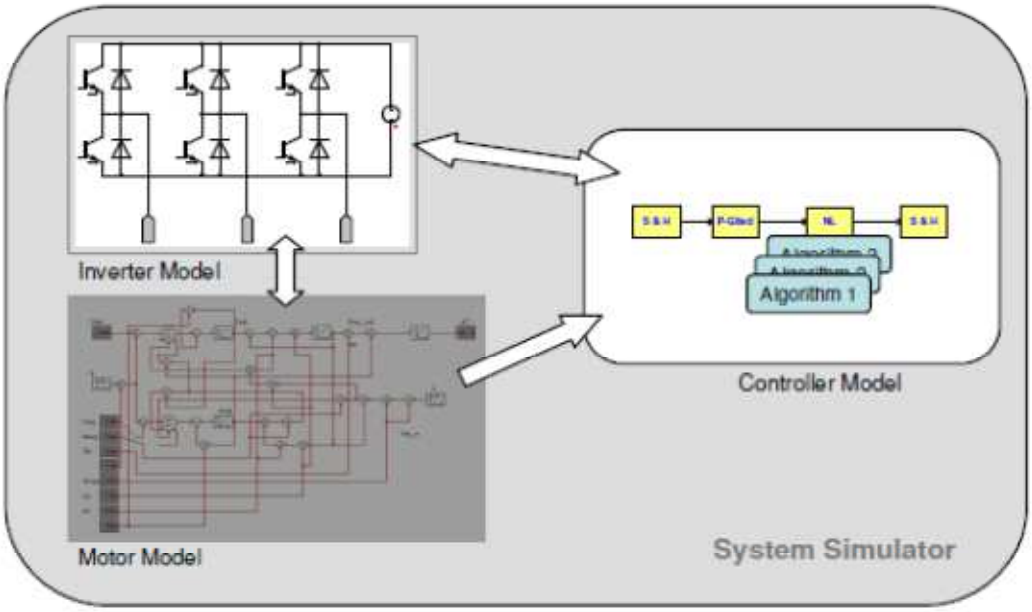


Abbildung 1: Software-in-the-loop

SIL ermöglicht den Test der erstellten Software vor der ersten Inbetriebnahme des Systems. Damit können Programmierfehler entdeckt werden, ehe sie evtl. zur Zerstörung von Komponenten führen. Dennoch ergeben sich mit diesem Ansatz und in Abhängigkeit von der Arbeitsweise des verwendeten Simulators eine Reihe von Einschränkungen:

· Die Abarbeitung des eingebundenen C-Codes wird in der Simulation einem diskreten Zeitpunkt zu geordnet. Verzögerungen zwischen Einlesen und Ausgeben von Signalen werden üblicherweise nicht berücksichtigt. Es können auch keine Aussagen darüber getroffen werden, ob die einzustellenden Abtastzeiten mit der verwendeten Zielplattform realisierbar sind.

·Es wird nur ein Teil des C-Codes eingelesen, da die Peripherie der Mikroprozessoren oftmals nur vereinfacht modelliert wird. Daraus ergibt sich die Notwendigkeit einer Code-Anpassung und u. U. die Vernachlässigung von Effekten, die im Zusammenhang mit AD- bzw. DA-Wandlungen stehen. (Barucki)

2.4 Hardware-in-the-loop

Nachdem die Struktur des Gesamtsystems einschließlich der verwendeten Steuer- und Regelalgorithmen feststeht, kann sich eine weitere Testphase anschließen. Die für die Regelungen zu verwendende realen Hardware (z. B. Mikrokontroller mit Peripherie) wird mit einem Simulationsmodell des übrigen Systems (z. B. Umrichter, Motor, mechanische Last) kombiniert. Dieses Simulationsmodell wird auf einer speziellen Hardware (üblicherweise DSP-basiert) in Echtzeit berechnet. Die Hardware liest die Signale der Ansteuerung ein und liefert Sensorsignale auf der Basis der Echtzeitsimulation zurück. Abbildung 5 zeigt die Struktur dieses Ansatzes, der als „Hardware-in-the-loop“ (HIL) bezeichnet wird. Er erlaubt den Test der kompletten Ansteuersoftware in der geplanten Zielplattform ohne das Risiko von Zerstörungen der Systemkomponenten. Die meisten Fehler in der erstellten Software können vor der ersten Inbetriebnahme erkannt werden, sofern das Modell des übrigen Systems realitätsnah genug ist.

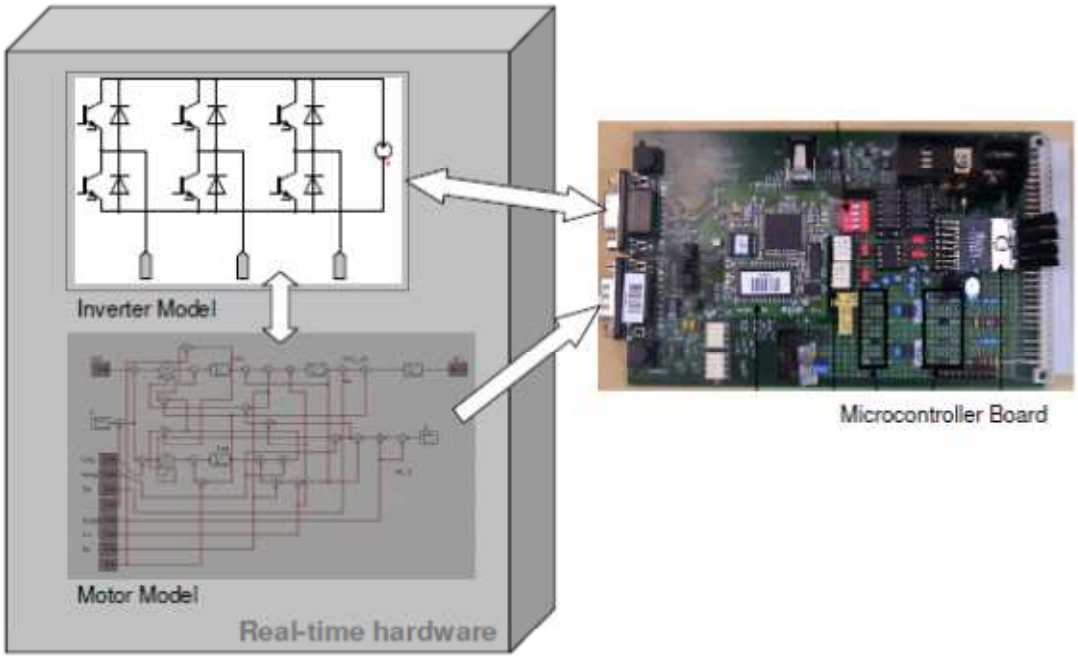


Abbildung 2: Hardware-in-the-loop

Der HIL-Ansatz ist mit folgenden Einschränkungen verbunden:

· Für die erforderliche Simulation des Systems in Echtzeit sind leistungsfähige und kostenintensive Prozessoren (DSPs) erforderlich, die als Steckkarten oder externe Geräte mit einem PC verbunden werden. In Verbindung mit der Zielhardware (Mikroprozessor mit Peripherie) ergibt sich damit ein Aufbau, der einer flexiblen Entwicklung im Wege steht. Die Arbeiten können nur am Versuchsaufbau vorgenommen werden.

· Zur Gewährleistung der Echtzeitfähigkeit sind im Allgemeinen nur Modelle mit geringer Abstraktionstiefe geeignet. Beispielsweise erfordert die Nachbildung hochfrequenter Schwingungen in leistungselektronischen Schaltungen Simulationsschrittweiten im Bereich von Nanosekunden oder darunter, so dass eine Echtzeitsimulation praktisch ausgeschlossen ist.

· Simulationsmodelle, die zu Algebro-Differential-Gleichungssystemen führen, können u. U. nicht eingesetzt werden. Durch den Verzicht auf derartige Modellstrukturen sind die Modellierungsmöglichkeiten eingeschränkt.

Die Debug-Möglichkeiten für den Software-Test in Echtzeit sind eingeschränkt. Zwar ist eine genaue Detektion potentieller Havariezustände

möglich, die Fehlersuche im Quellcode muss jedoch separat erfolgen. (Barucki)

1.5 Rapid Control Prototyping(RCP) und Hardware in loope (HIL)Phasen

Der Entwurfsprozess einer bestimmten Funktion im regelungstechnischen Sinne erfolgt grundsätzlich in zwei Entwicklungsphasen. Mit der Rapid Control Prototyping (RCP) Phase werden Methoden bzw. Werkzeuge bezeichnet, die zum schnellen und effizienten Bearbeiten von regelungstechnischen Problemstellungen dienen. Das beinhaltet (1) die theoretischen Methoden der Regelungstechnik von der Modellbildung und der Systemanalyse bis zum Entwurf eines Regelungskonzeptes, (2) den Test und die Voroptimierung der Regelung in der Simulation, (3) die automatische Code-Erzeugung für einen Echtzeitrechner innerhalb einer realen Versuchsumgebung und (4) die Verifikation und Optimierung der Regelung mit dem zu regelnden Objekt auf dem Versuchsstand. Das Ergebnis ist dann ein fertiger Regler-Prototyp der auf einem Steuergerät implementiert und optimiert werden muss. Ein zuverlässiger und kosteneffizienter Test von Steuergeräten ist nur in einem geschlossenen Regelkreis möglich, wobei dem Steuergerät eine möglichst realitätsnahe Simulationsumgebung bereitgestellt wird. Das Steuergerät soll also identische Signale „sehen“ mit denen es in einem echten Fahrzeugeinsatz interagiert. Diese Wechselwirkung zwischen dem Steuergerät und der Simulationsumgebung stellt die zweite Entwicklungsphase dar und wird als Hardware-in-the-Loop (HIL)-Simulation bezeichnet. Ein Vorteil dieses Ansatzes liegt vor allem an den umfangreichen Test- und Optimierungsmöglichkeiten des zu entwerfenden Systems ohne direkte Präsenz der echten physikalischen Regelstrecke. Die in der Realität nicht oder nur schwer messbaren Zustände der Regelstrecke sind am HIL-Simulator einfach zugänglich. Ein weiterer Vorteil ist die Möglichkeit, Steuergeräte in automatisierten Testprozeduren zu testen bzw. optimieren. (N.Bajcinca)

1.6 Serienfertigung

In der Serienfertigung werden bestimmte Stammdaten zur Verfügung gestellt, mit denen die Serienfertigung durchgeführt wird. Zu den Stammdaten gehören u.a. das Serienfertigungsprofil und der Produktkostensammler.

- Planungstableau

Die Planung im Rahmen der Serienfertigung erfolgt periodenbezogen. So können Sie, ausgehend von der vorliegenden Bedarfssituation, periodenbezogene Produktionsmengen einplanen. Die Produktionspläne für Erzeugnisse und Erzeugnis Gruppen werden periodenbezogen aufbereitet und dem Benutzer in einer solchen periodischen Sicht zur Kontrolle und Bearbeitung angeboten.

- Sequenzplanung

Mittels der Sequenzplanung können Sie eine Taktterminierung durchführen und dabei die Reihenfolge bestimmen, in der Sie die Planaufträge auf der Linie fertigen. Sie können insbesondere hohe Auftragsvolumen leichter einplanen und grafisch darstellen.

- Materialbereitstellungsliste

Mittels der Materialbereitstellungsliste können Sie den innerbetrieblichen Materialfluß zur Versorgung der Produktion steuern. Die Materialbereitstellungsliste überprüft die Bestandssituation an der Linie, errechnet die Fehlmengen für die Komponenten und stößt für die Fehlmengen den Nachschub an.

- Ist-Datenerfassung

Die Rückmeldung des Produktionsfortschritts erfolgt vereinfacht mit Bezug auf die gefertigten Mengen und schließt normalerweise die retrograde Entnahmebuchung von Komponenten und das Buchen von Fertigungskosten ein.

- Kostenträgerrechnung

In der Serienfertigung rechnen Sie in der Regel die Kosten materialgenau oder versionsgenau über einen Produktkostensammler ab (periodisches Produkt-Controlling). (SAP AG)

3.Hardware

3.1 .FLEX-Boards

Die FLEX-Boards sind eine Familie von Entwicklungs-Boards, welche vor allem für pädagogische Zwecke und schnelles Prototyping geeignet sind.

Seit ein paar Jahren sind sie auf einer gemeinsame Plattform von mehr als 15 Universitäten und einer Reihe von Unternehmen für den Unterricht von Prototyping und Sensor-Schnittstellen verwendet worden.

Es folgt eine kurze Beschreibung der FLEX Boards .

3.1.1 a) :FLEX- Full und b) :FLEX- Light

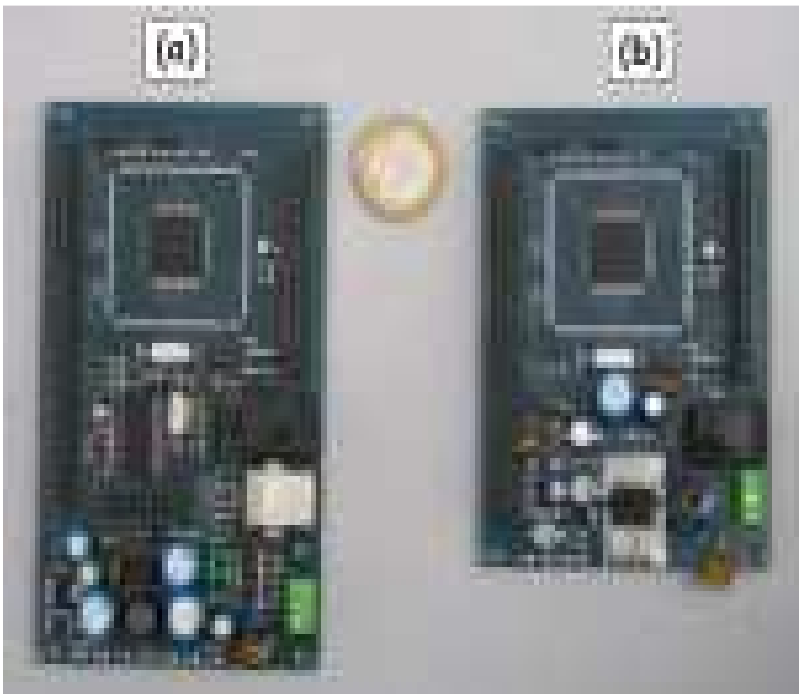


Abbildung 3: Flex -Full und Felex-light-Base

Flex-light-base-Schema

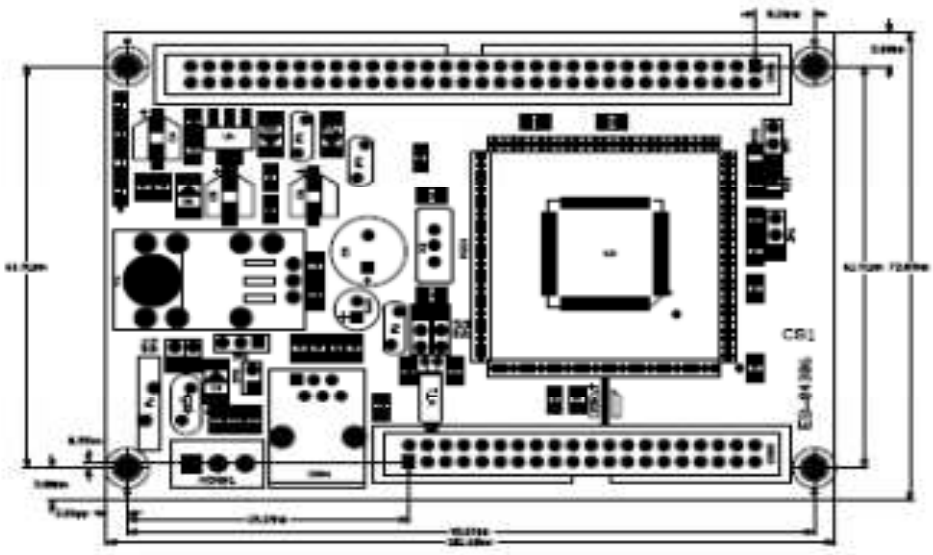


Abbildung 4: Felex-light-Base-Schema

Dies sind die Basis-Boards und bieten einen Microchip dsPIC Mikrocontroller. Alle Pins sind auf einem 2,54 mm-Stecker, für eine einfache Verbindung mit einem Durchgangsloch Bord .

Die **Flex-Light** hostet die dsPIC mit einem linearen Spannungsregler, während im Falle der **FLEX- Full**, ein Schaltnetzteil Lieferanten sowie eine USB-Verbindung mit einem PIC18.

3.1.2 FLEX Demo Board

Abbildung 5: Flex -Demo Board

Die **Demo Board** ist ein Tocheter Bord, das auf der Oberseite des FLEX- Full und Flex-Light eingesteckt werden kann.

Es enthält Schaltflächen, LEDs, ein 16x2 LCD, 3-Achsen-Beschleunigungssensor, IR TX / RX, 802.15.4-Slot (für Chipcon oder Microchip-Transceiver), 2 DAC, Potentiometer, Lichtsensor, Temperaturfühler sowie eine serienmäßige Schnittstelle.

3.1.3 FLEX Demo2 board / pack

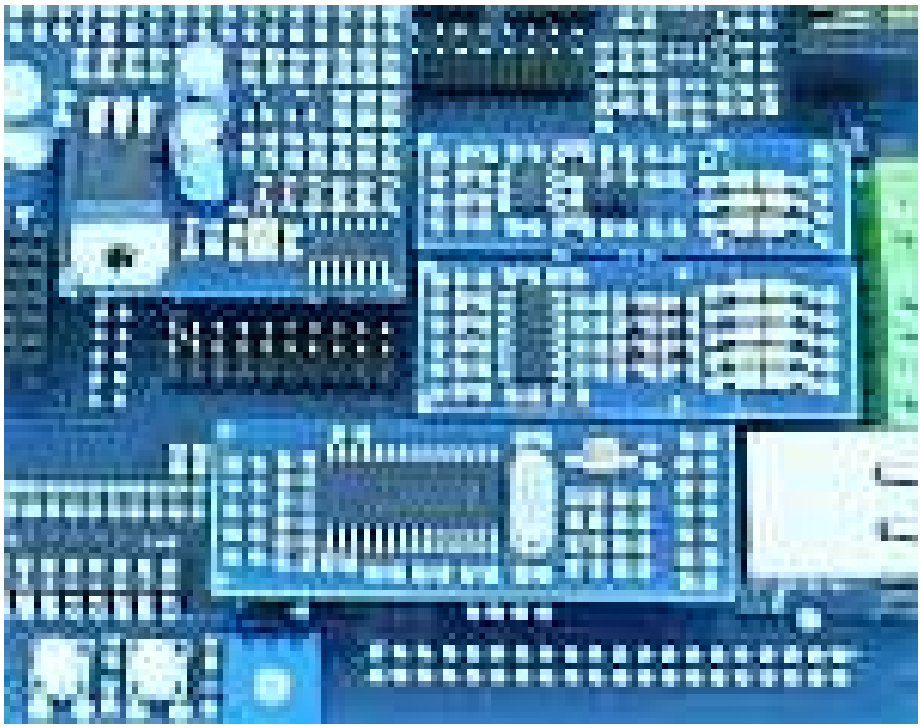


Abbildung 6: Flex -Full und Felex-light

Das **Demo2 board / pack** ist ein Tochter Bord, das auf der Oberseite des **Flex-Light** und **FLEX- Full** eingesteckt werden kann.

Das FLEX demo2 Pack, auch als FLEX Motion-Vorstands-Pack bezeichnet, ist ein FLEX Daughter-Board mit Plug-Ins, welches speziell für Bildungseinrichtungen wie z.B. Schulen und Universitäten verwendet wird .

3.1.4 FLEX Multibus Bord

Abbildung 7: Multibus Board

Das **Multibus Board** ist ein Daughter Bord, das auf der Oberseite des **FLEX- Full** und **Flex-Light** eingesteckt werden kann.

Es verfügt über 2 serienmäßige Slots, 2 CAN-Steckplätze, 1 SPI-Slot und 1 Ethernet-Steckplatz.

Diese sind eine Reihe von kleinen Modulen, die auf der Oberseite des **Demo Board / Multibus Boards** eingesteckt werden können .

4. Installation von Scicoslab-4.4.1 und Erika Enterprise

4.1 Aufbau und Installationsschritte zum Einrichten der Scicoslab Code-Generierung für FLEX

4.1.1 Aufbauschema:

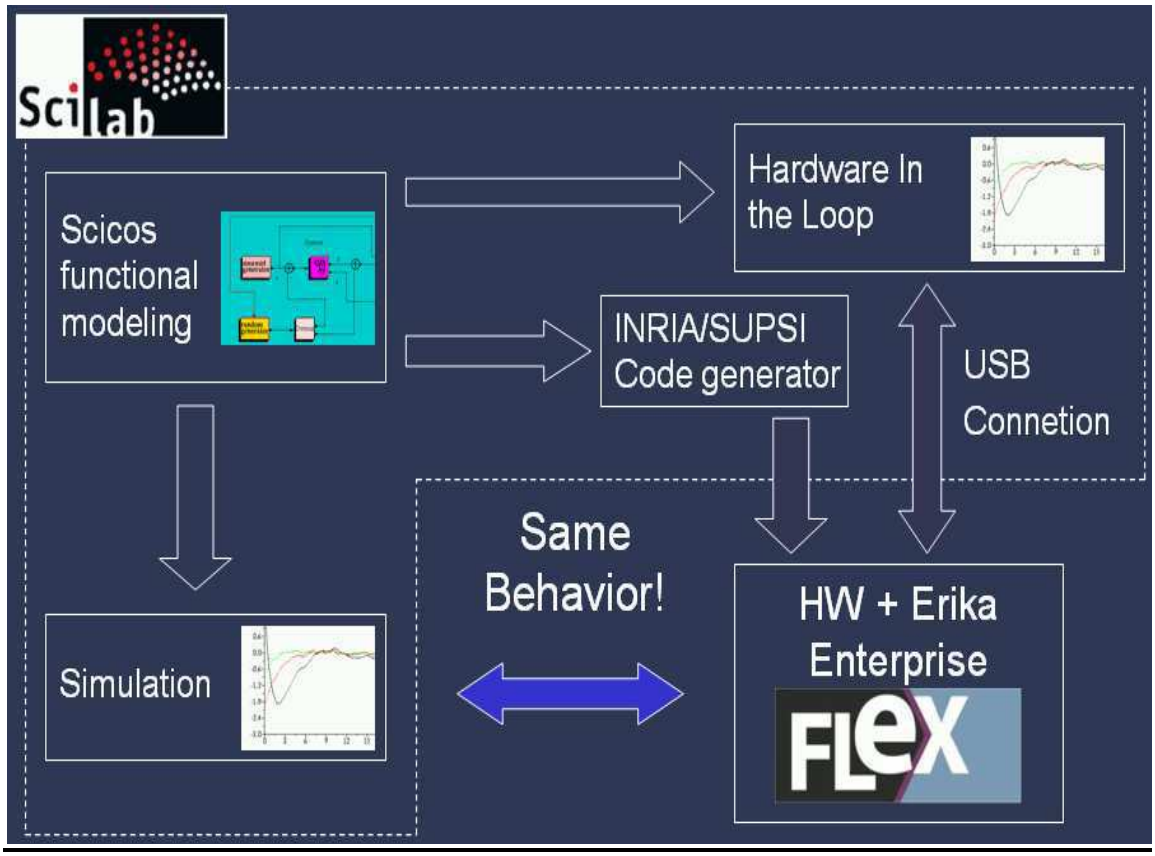


Abbildung 8 : Aufbau Schema

4.1.2 Installationsschritte

1. Bevor Sie mit der Installation beginnen ,entfernen Sie bitte die alte Version von Erika Enterprise.
2. Laden Sie die neue Version von Microchip MPLAB IDE von der MPLAB IDE Website herunter und installieren Sie diese. Dies ist erforderlich um die dsPIC Mikrocontroller auf dem FLEX programmieren zu können.
3. Laden Sie einen C30-Compiler von der Microchip MPLAB C30 Compiler Website herunter und installieren Sie diesen. Ein Compiler wird benötigt, um Ihre Anwendung zu kompilieren zu können.
4. Downloaden und installieren Sie Cygwin von der Cygwin Website. Cygwin ist erforderlich, um den Code aus Ihrem Scicos erstellte Diagramm erstellen zu können. (siehe Abbildung 9)

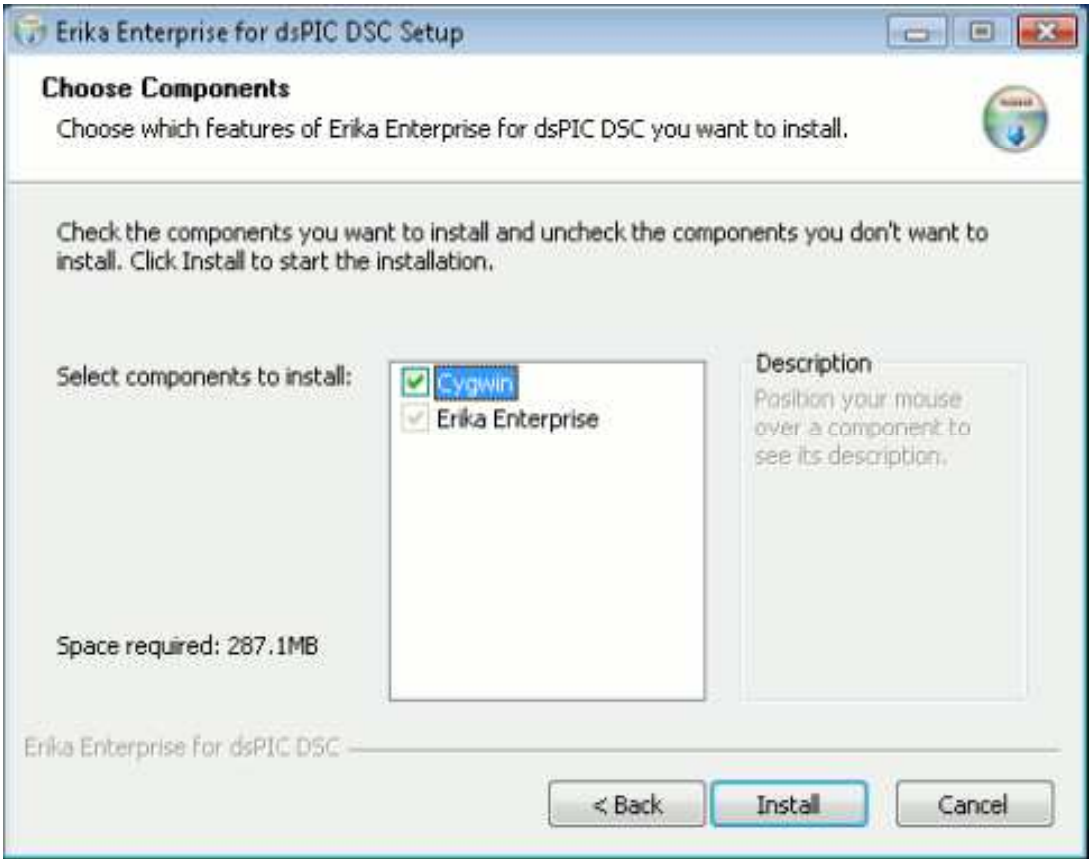


Abbildung 9: Dieses Fenster zeigt das Dialogfeldmit den verfügbaren Installationspakete

Das Installationsprogramm wird nach einem Zielverzeichnis fragen.
Wenn möglich nutzen Sie bitte c: / Verzeichnis
(siehe Abbildung 10).

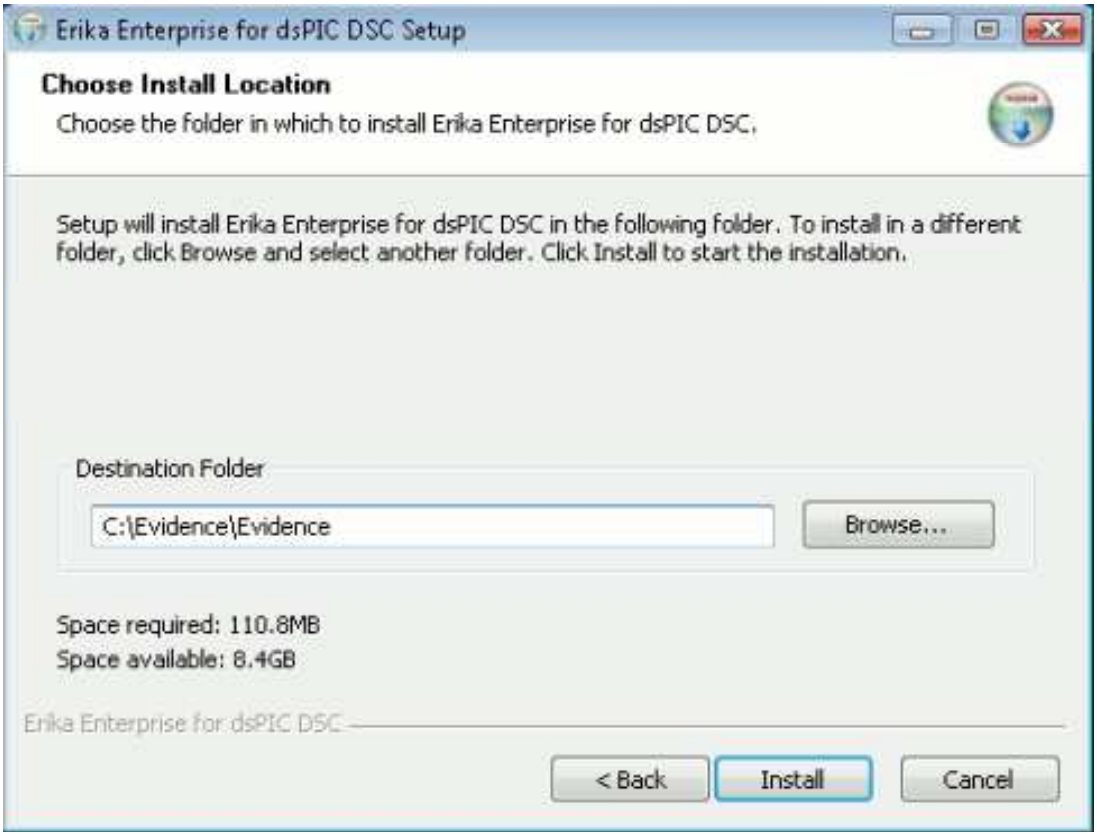


Abbildung 10: Destination dir für die Installation von Erika Enterprise

An diesem Punkt, überprüfen Sie bitte die erste Zeile der Datei evidencedir \ bin \ mymake_cygwin.bat
Zum Beispiel, wenn Cygwin in C installiert ist: \ cygwin, dann sollte die erste Zeile der Datei wie folgt aussehen:

```
@set EE_BASH_PATH=C:\ cygwin \ bin\ bash
```

5. Downloaden von Microsoft Visual C + + 2008 von der Microsoft Visual C + + site und installieren Sie es. Es ist das Evidence Scicoslab Pack erforderlich.

6. Laden Sie Sie die neueste Scicoslab_pack von der Homepage www. http://erika.tuxfamily.org herunter. Entpacken Sie das Paket und installieren Sie es.
7. Bitte überprüfen Sie die Datei, die (c:/Evidence/Evidence/bin/rtd_config.properties) enthält .sinnvolle Einstellungen für Ihre Installation. Mögliche Einstellungen sind die Kommentare in der Datei-Eigenschaften. Insbesondere sollten Sie die richtigen Lage des Microchip C30 und der ASM30 Werkzeuge angeben. Das folgende ist ein Beispiel für eine italienische Installation :

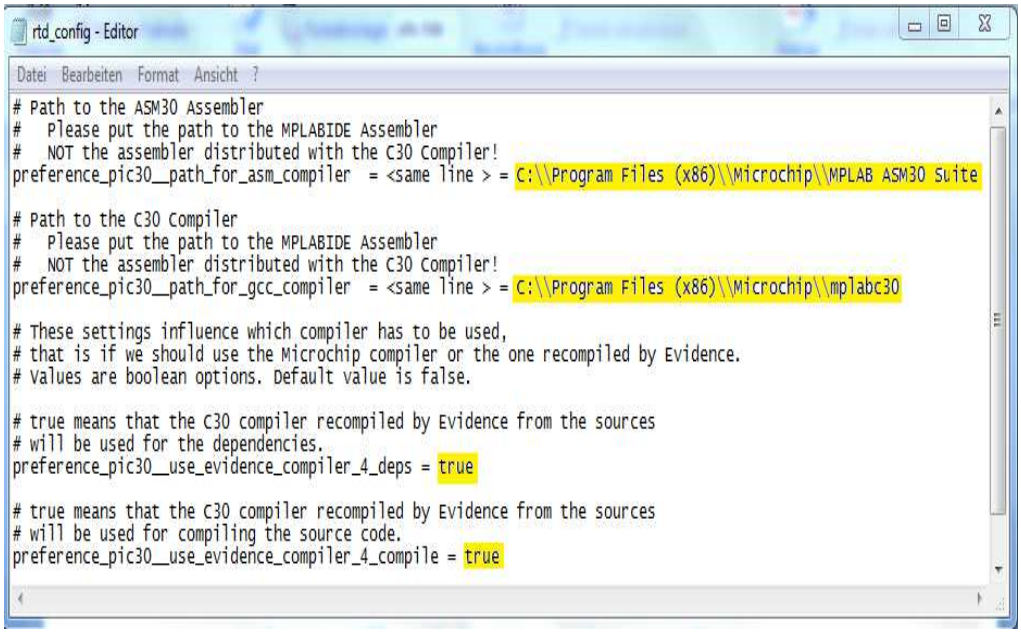
   preference_pic30__path_for_asm_compiler = <same line >
   C :\\ Programme\\ Microchip\\ MPLAB ASM30 Suite
   preference_pic30__path_for_gcc_compiler = <same line >
   C :\\ Programme\\ Microchip\\ MPLAB C30

   preference_pic30__use_evidence_compiler_4_deps = true

   preference_pic30__use_evidence_compiler_4_compile = true

   Hinweis: Bitte beachten Sie einen doppelten Backslash und nicht einen einzelnen Backslash zu verwenden.

   Die folgende Abbildung zeigt die sinnvolle Einstellungen für Ihre Installation



8. Laden Sie ScicosLab- 4.4.1 aus dem www. http://erika.tuxfamily.org herunter und installieren Sie es in C:/Evidence/scilab-4.4.1.Bitte

verwenden Sie keine Leerzeichen. Am Ende der Installation starten ScicosLab neu, damit die Änderungen wirksam werden.

9. Löschen Sie das **contrib-Verzeichnis** in Ihrer Scilab-Installation, und ersetzen Sie es mit dem **scicos_ee** Verzeichnis, das Sie entpackt haben. Dieser Schritt kopiert den Code-Generator und die Paletten für FLEX in das Scicos Installationsverzeichnis.

4.1.3. Erstellen und kompilieren den ersten Scicos-Diagramm

1. Bitte starten Sie ScicosLab aus dem Startmenü.
2. Im Scicoslab-Menü Applications ⇒ Scicos auswählen oder Befehl Scicos () auf der Scicoslab-Oberfläche eingeben und Enter drücken . Abbildung 11

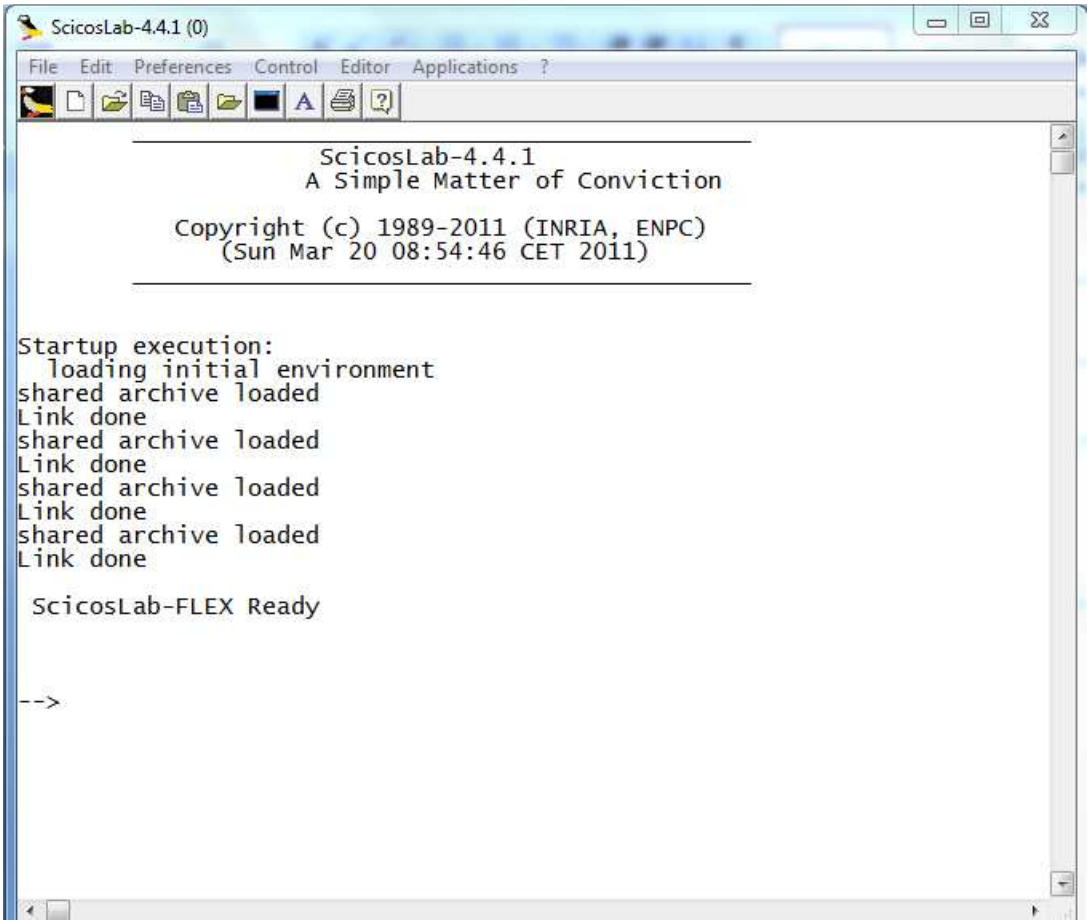


Abbildung 11: Die Scicoslab-Oberfläche

3. Die Scicos Fenster erscheint, wie gezeigt in Abbildung 3.2

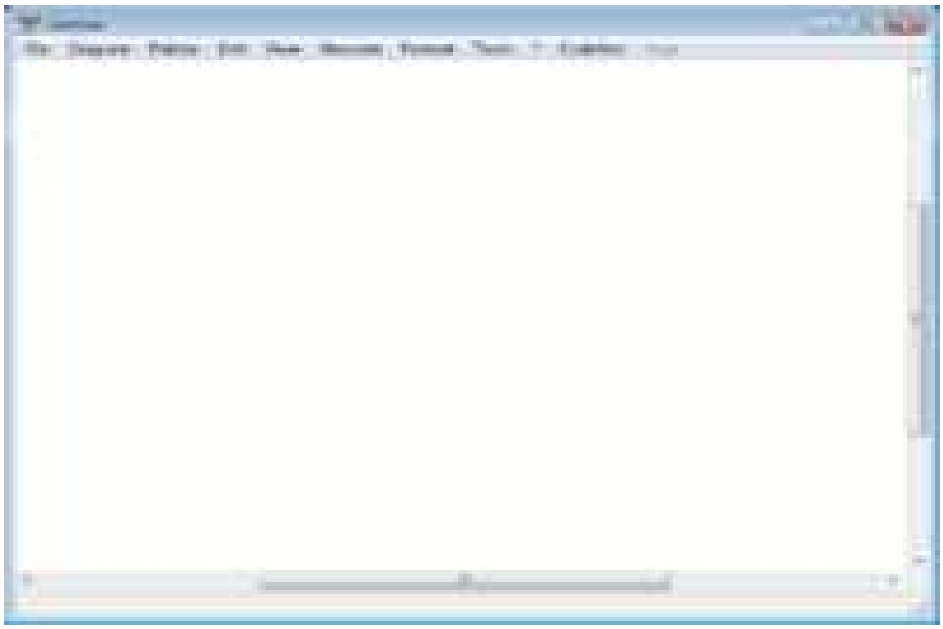


Abbildung 12: Die Scicos Splash-Screen

4. Wählen Sie Paletten aus dem Paletten-Menü, wie in Abbildung 13 angezeigt

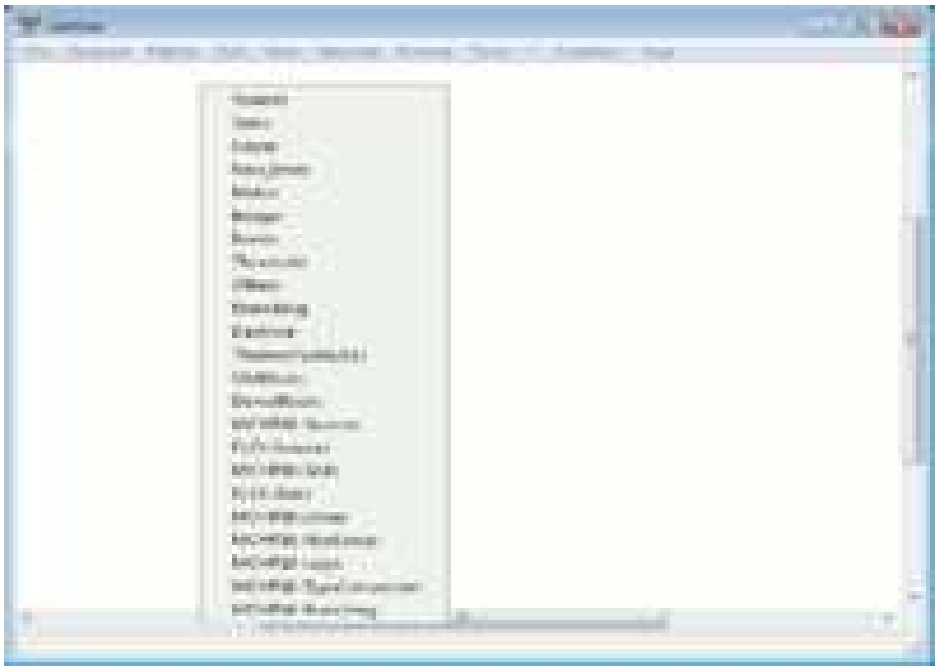


Abbildung 13 : Die Palette.

5. Eine kleine Liste erscheint anstelle des Menüs. Wählen Sie *FLEX,* wie gezeigt, in Abbildung 14

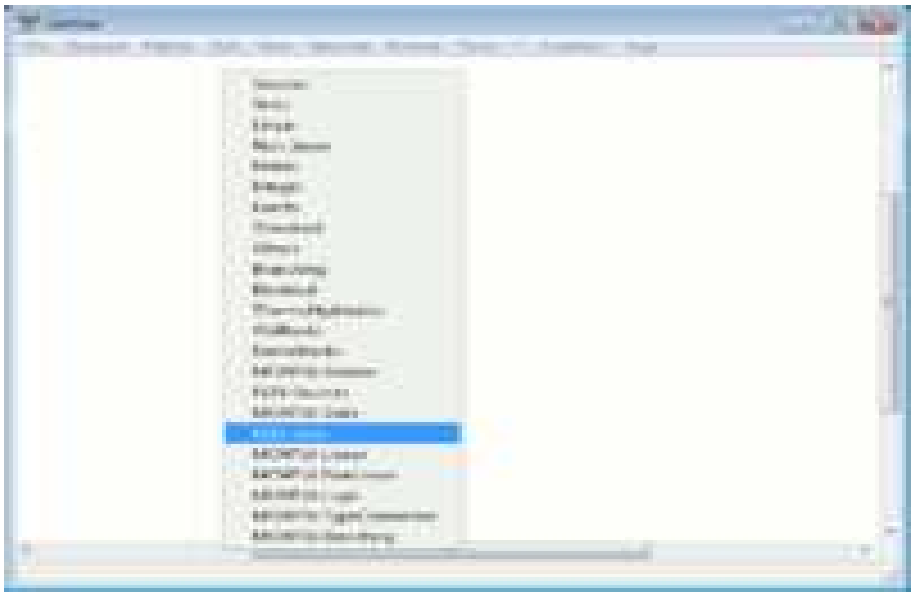


Abbildung 14: Die Palette Liste

6. Ein Fenster erscheint, mit einigen Blöcke für die FLEX-Boards , wie gezeigt, in Abbildung 15

Abbildung 15: Die dsPIC Palette

Klicken Sie einmal auf den FLEX-LED-Block. Das Fenster Auswahl erscheint auf der Scicos Oberfläche. Die Maus wird nun ein weißes Rechteck von der Dimension des LED-Block. Klicken Sie in einen Teil des

weißen Fensters. Ein LED-Block wurde dem Diagramm hinzugefügt, wie die Abbildung 16 zeigt.

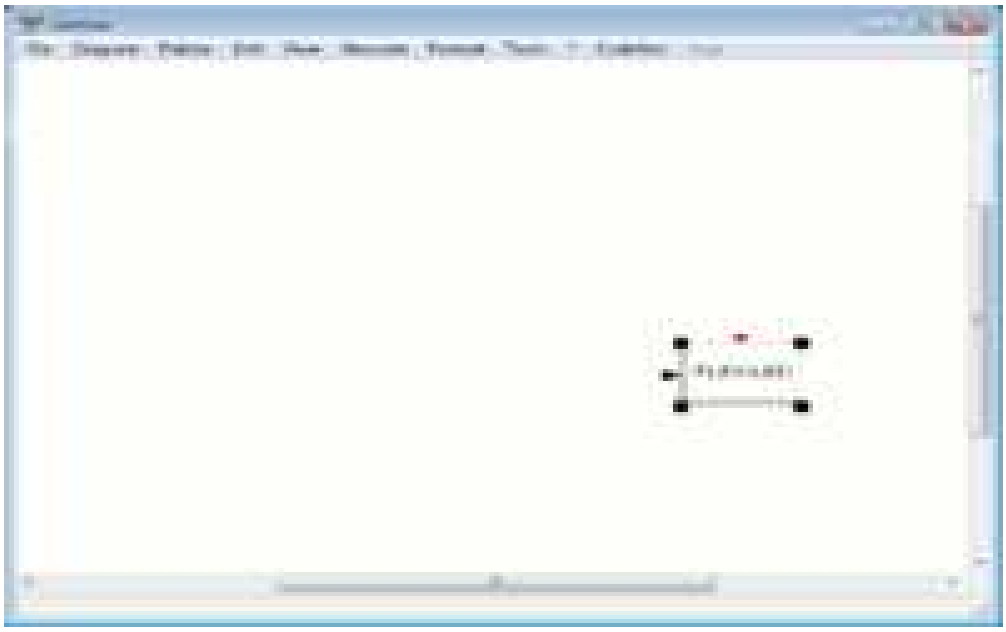


Abbildung 16: Die LED-Block ist in dasDesign-Fenster fiel

- **Hinweis:** Wenn Sie einen Block verschieben müssen, gehen Sie mit der Maus auf den Block und drücken Sie m, dann bewegen Sie den Block und klicken Sie auf die neue Position.
- **Hinweis:** Wenn Sie einen Block oder eine Zeile löschen müssen, wählen Sie den Block, und drücken Sie d.

8. Öffnen Sie die MCHP16-Sources-Palette, und wiederholen Sie das gleiche mit dem Sinus-Block und platzieren Sie diesen auf der linken Seite des LED-Block, wie in Abbildung 17

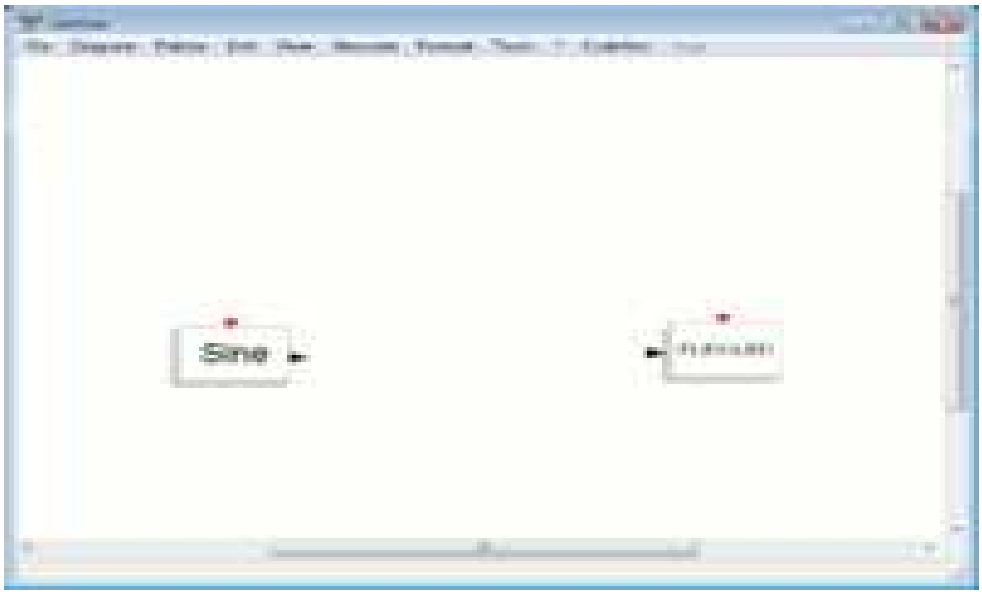


Abbildung 17: die Sine-Block auf der linken Seite des LED-Block

9. Um die beiden Blöcke zu verbinden, gehen Sie mit der Maus auf das schwarze Dreieck des Sine-Block und drücken Sie L(oder mit der linken Maustaste anklicken) ,dann auf das schwarze Dreieck des LED-Block klicken. (Siehe Abbildung 18)

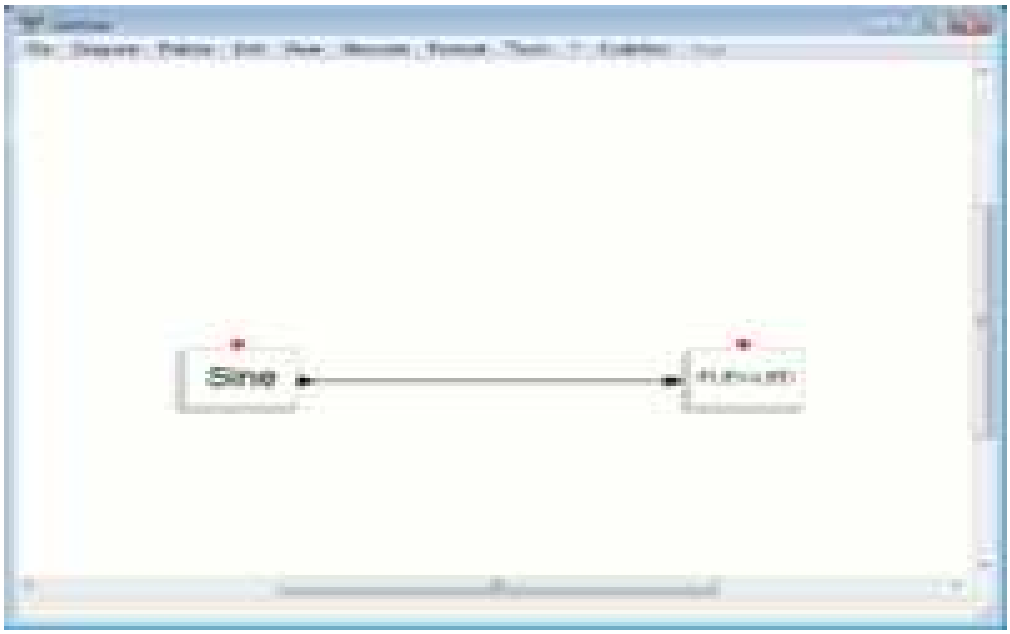


Abbildung 18: Sine und LED sind nun miteinander verbunden

10. Wählen Sie die rote Uhr aus der MCHP16-Sources-Palette und fügen Sie es dem Diagramm hinzu, wie in Abbildung 19

Abbildung 19: Der Clock-Block über dem Sinus-und LED-Block

11. Verbinden Sie nun das Taktsignal der Uhr mit den beiden Blöcken. Um das zu tun genügt einfacher Klick auf das rote Dreieck des Clock-Blockes, dann unter den Clock-Block klicken, dann Klick auf den Sinus-Block, dann auf das rote Dreieck des Sinus-Block klicken. Danach ein einfacher Klick auf die Zeile unter des Clock-Block. Drücken Sie die Taste 'L' auf der Tastatur, dann über den LED-Block, dann auf das rote Dreieck des LED-Block. Das Ergebnis ist in Abbildung 20

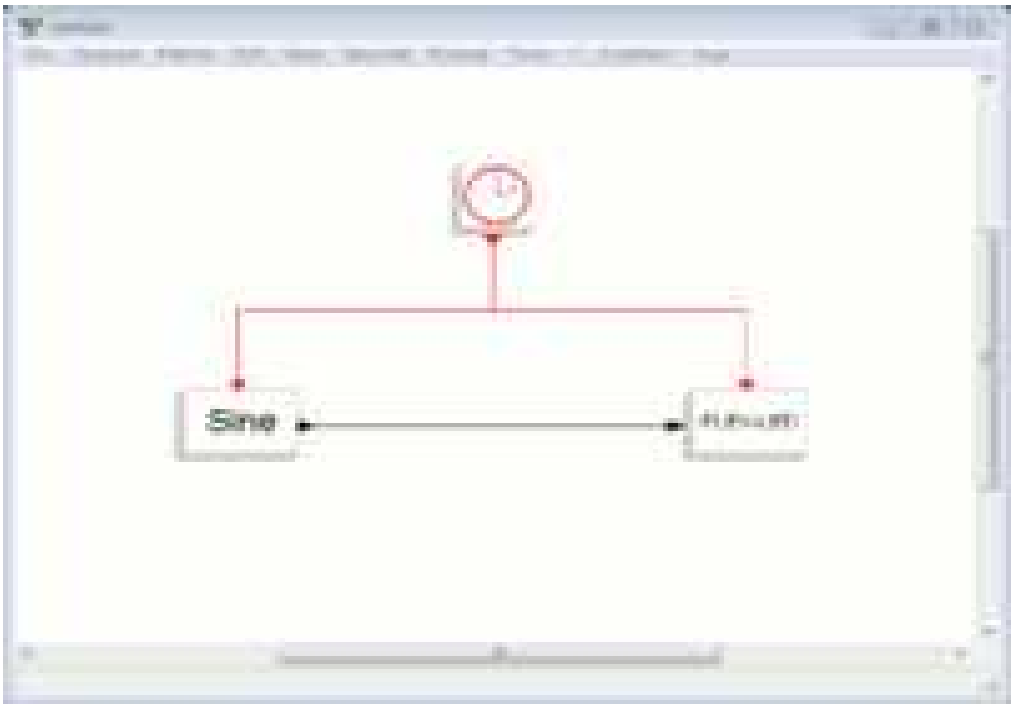


Abbildung 20: Der Clock-Block ist mit dem Sinus-und LED-Block verbunden

12. Klicken Sie einmal auf die Clock-Block. Ein Eigenschaften-Fenster erscheint. Lassen Sie dies unberührt und drücken Sie OK. Sie können das gleiche mit dem Sine-Block tun. Die beiden folgenden Abbildungen 21 und 22 zeigen diese Fenster.

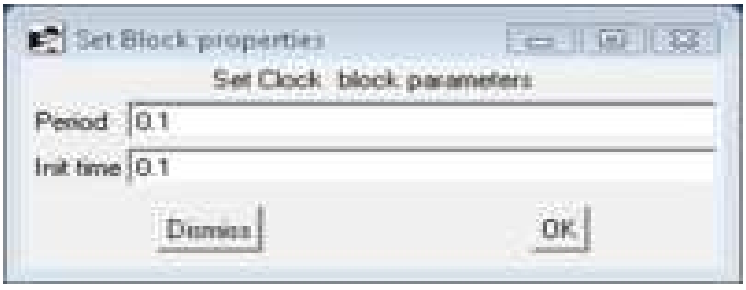


Abbildung 21: Die Clock-Block Eigenschaften

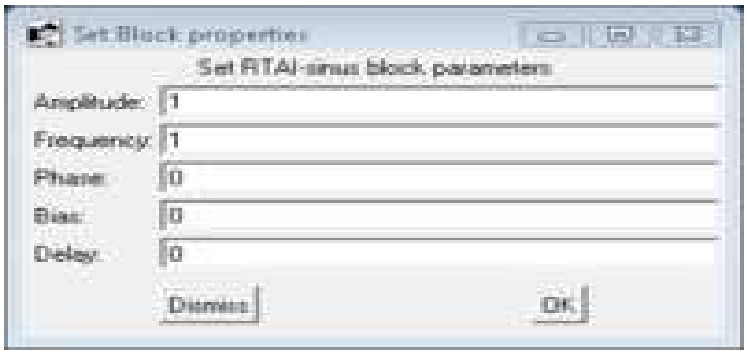


Abbildung 22: Die Sinus-Block Eigenschaften

13. Der Code-Generator kann einen Code erzeugen der nur von einem speziellen Baustein mit dem Namen *Super Block* stammen kann. Aus diesem Grund müssen wir ein Super Block erstellen der den Sinus- und der LED-Blöcke umschließt. Um dies zu erreichen, wählen Sie die den Menüpunkt *Region to Super Block* aus dem Menü *Diagramm* aus. (siehe Abbildung 23).

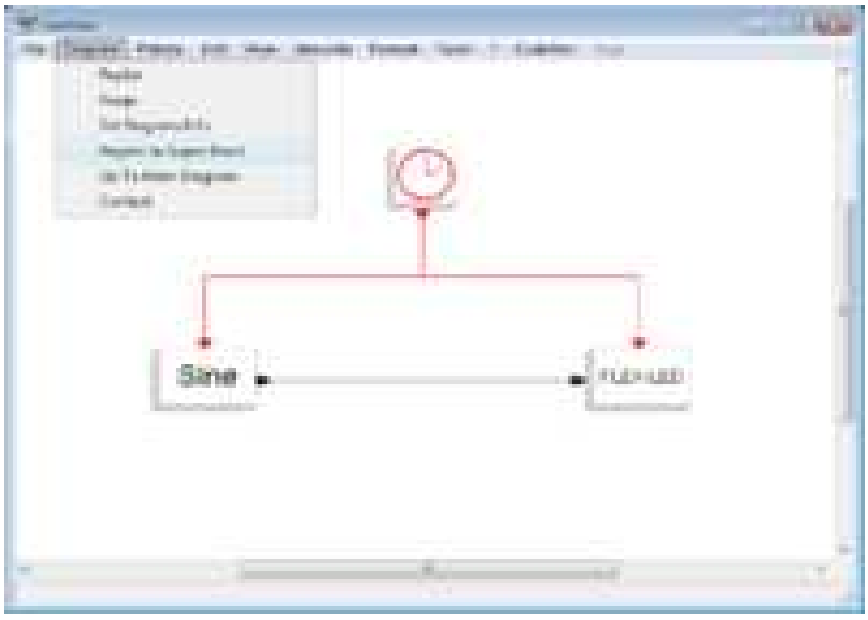


Abbildung 23: Die Region im Super Block Menüpunkt

14. Dann treffen Sie eine Auswahl , dass der Sinus-Block, der LED-Block und die roten Linien in einer Weise zusammengeführt werden, dass nur *eine* rote Linie existiert, wie in Abbildung 24 enthält

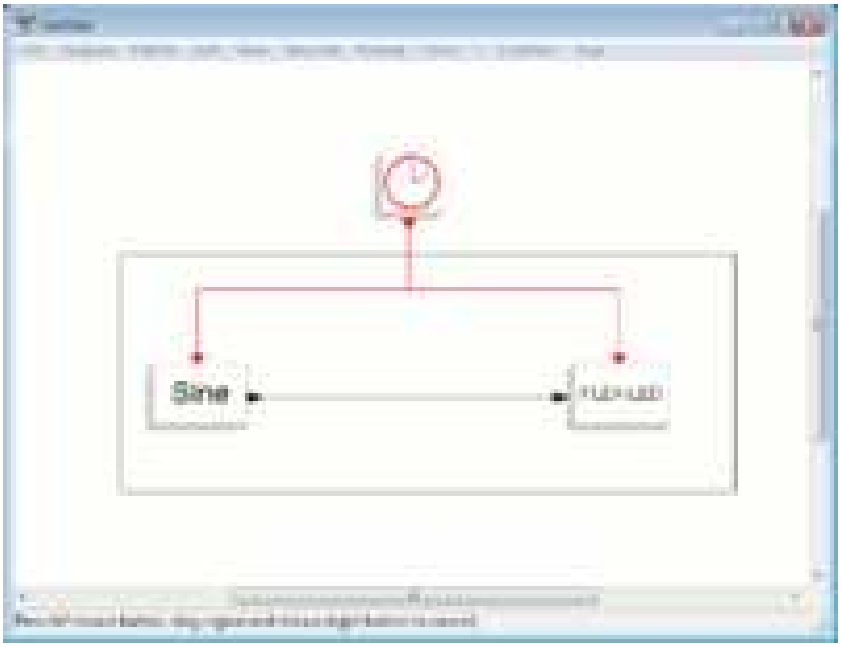


Abbildung 24: Auswahl um einem Super Block zu schaffen

15. Als Ergebnis ist ein Super Block (siehe Abbildung 25)

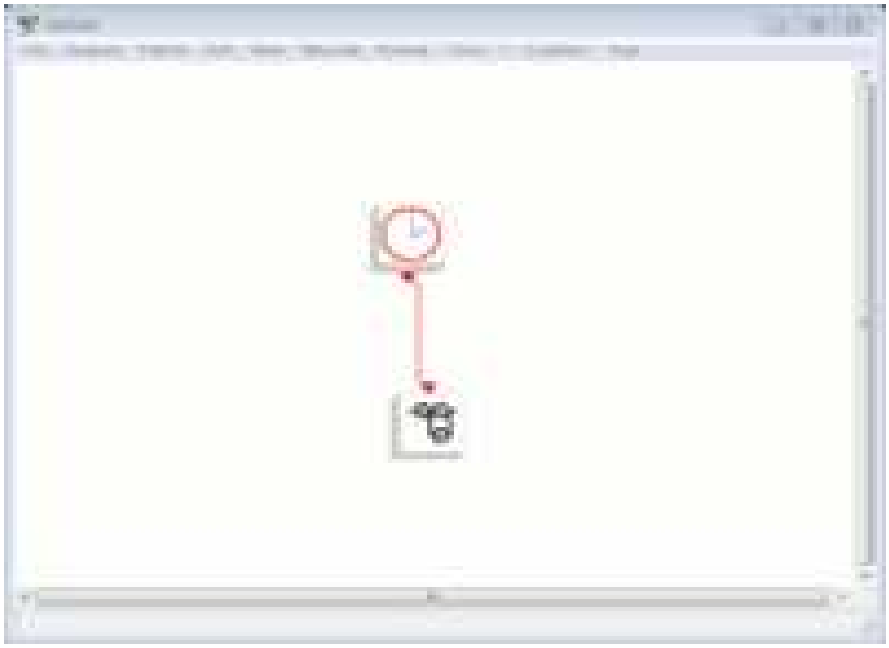


Abbildung 25: Der Super Block

Welche den Sinus-und LED-Block enthält. Um diese Blöcke zu sehen genügt ein einzelner Klick auf den Super Block und ein weiteres Fenster wird angezeigt (siehe Abbildung 26).

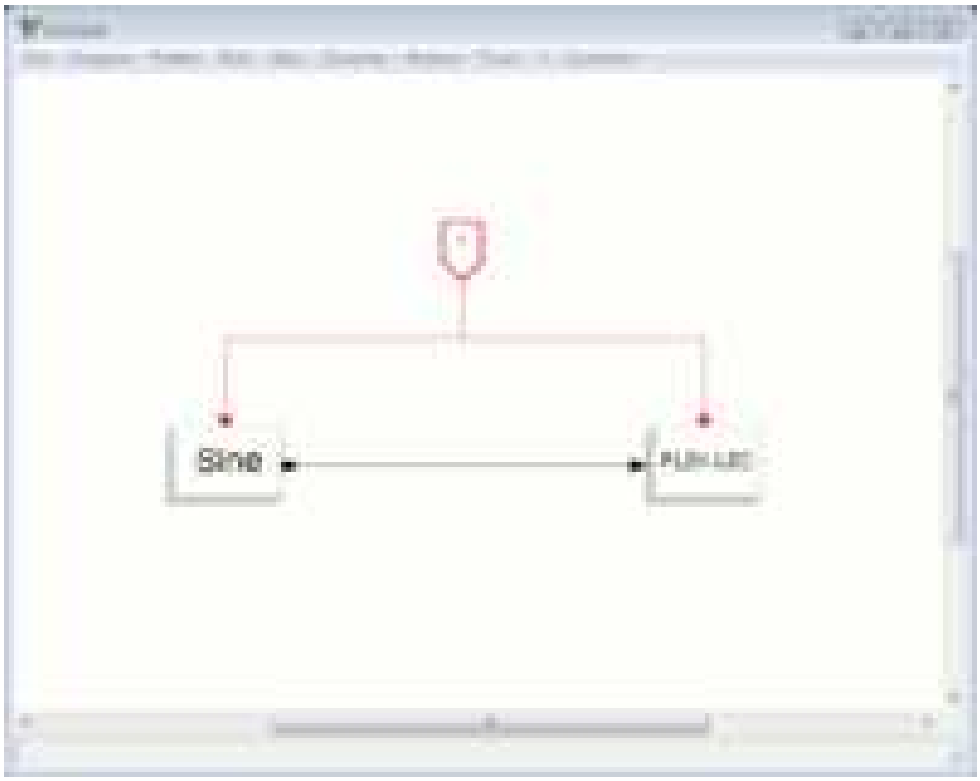


Abbildung 26: Der Inhalt des Super Block

16. Bitte beachten Sie, dass dieses Fenster sehr ähnlich zu den vorherigen ist, mit der Ausnahme, dass das Objekt der Uhr Platzhalter mit der Nummer 1 substituiert ist.

17. **Hinweis:** Das Diagramm mit den Super Block ist deaktiviert, wenn das Super Block Diagramm angezeigt wird. Nur ein Fenster kann in einer Zeit, in Scicos aktiviert sein. Es ist nun Zeit, um die beiden Diagramme zu speichern. In dem Menü *Datei auf Speichern unter.* Speichern Sie die Grafik mit dem Super Block als led_sin.cos.

4.1.4 Generieren des dsPIC-Code aus einem Scicos- Diagramm

Es ist jetzt Zeit, um den Code für das Beispiel ,dass wir gerade erstellt haben ,zu generieren.

Hinweis: Eine Kopie der Datei in den vorherigen Schritten wurde innerhalb der scicos_examples / led_sin Verzeichnis aufgenommen. Um es zu öffnen, Doppelklick auf dieDatei scicos_examples / led_sin / led_sin.cos .

1. Wählen Sie Set Target aus dem CodeGen Menü aus (siehe Abbildung 27).

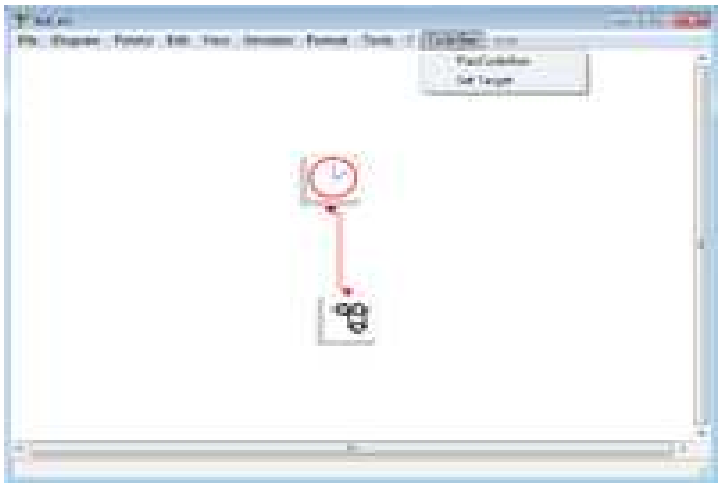


Abbildung 27: Das CodeGen Menü - Target

2. Ein Fenster erscheint, wie in Abbildung 28

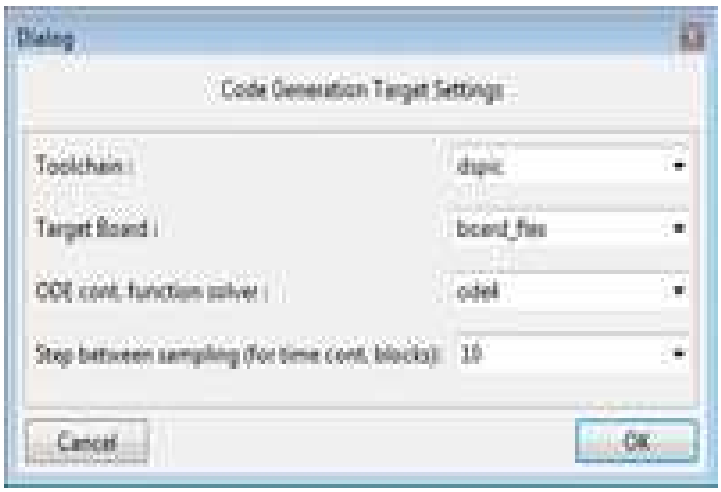


Abbildung 28: Das setTarget Dialogfeld

3. Sie können die Target-Board (board_flex oder board_easylab) mit dem zweiten Textfeld spezifizieren. Bitte lassen Sie die anderen Optionen unverändert.
4. Wählen Sie *FlexCodeGen* aus dem *CodeGen* Menü (siehe Abbildung 3.19).

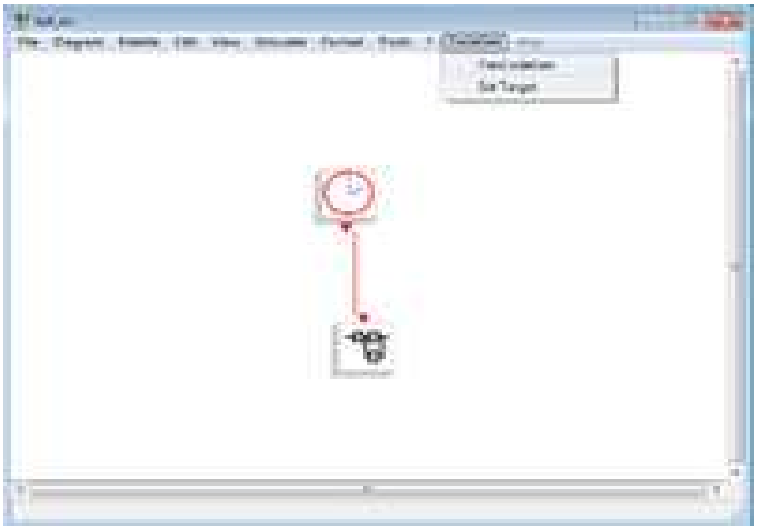


Abbildung 29: Die CodeGen Menü - Flex-Code-Generator

5. Ein Fenster erscheint, wie in Abbildung

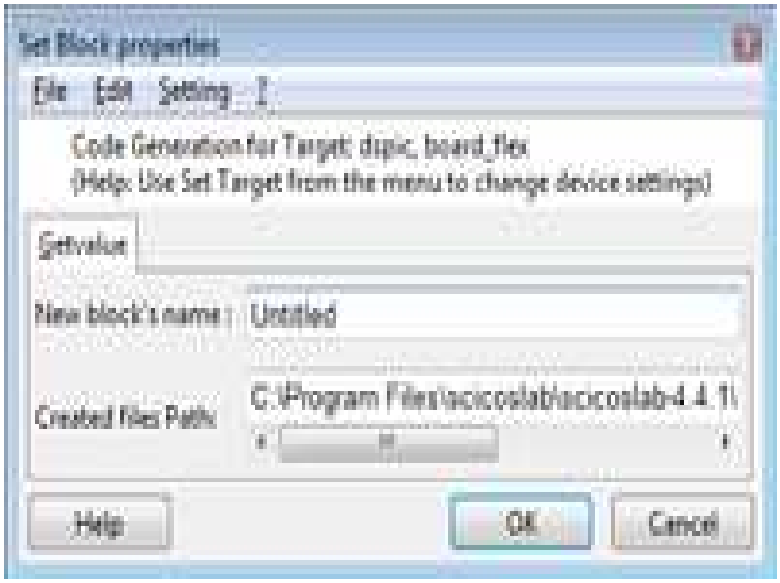


Abbildung 30: Das FlexCodeGen Dialogfeld

6. Sie können den Block Namen spezifizieren, indem Sie die " New block's name textbox" und das Verzeichnis, wo alle erstellten Dateien modifiziert werden soll.
    - **WICHTIGER HINWEIS für Windows-Benutzer:** Wählen Sie einen Ort, an dem Sie Schreibrechte haben. Wenn Sie ScicosLab mit Administrator-Rechten starten gibt es keine Einschränkungen in der Wahl der Zielordner.
7. Drücken Sie *OK.* Als Ergebnis sind eine Reihe von Dateien in das Output-Verzeichnis hinzugefügt worden.
8. Scicos öffnet automatisch ein Konsolen-Fenster, im Inneren laufen die folgenden Befehle:
    - die RT-Druid Vorlage-Generator an den Scicos Vorlage Anwendung instanziieren;
    - die RT-Druid Standalone-Code-Generator, um die ERIKA Enterprise-Konfigurationsdateien aus dem generierten OIL-Datei zu erzeugen;
    - die sie entweder an den Code zu kompilieren.
9. Das Ergebnis der Codegenerierung ist in Abbildung 3.21 dargestellt.

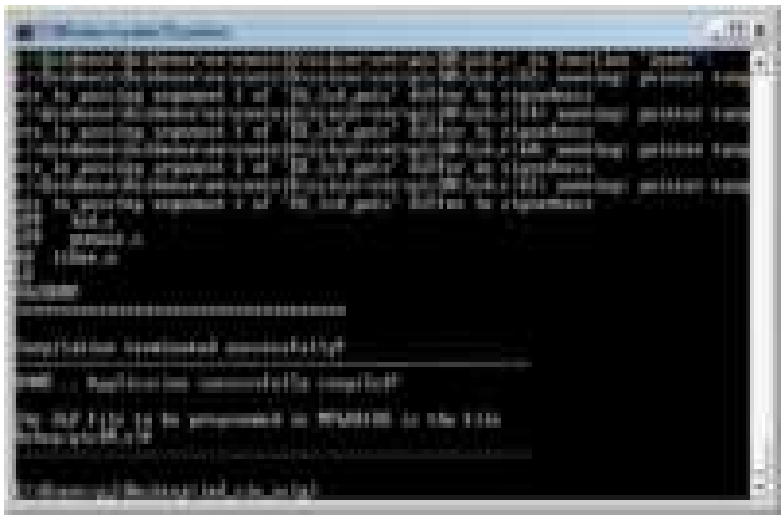


Abbildung 31: Die Zusammenstellung Konsole.

10. Die ausführbare Datei heißt **pic30**.elf und wurde in dem Debug-Verzeichnis wie gewohnt für alle ERIKA Enterprise-Anwendungen entfernt.
11. Jetzt können Sie Ihre Anwendung auf Ihrem FLEX Bord programmieren. Um dies zu erreichen, müssen Sie die MPLABIDE öffnen um wie gewohnt andere ERIKA Enterprise-Anwendungen zu programmieren. Bitte beachten Sie die ERIKA Enterprise-Tutorial für dsPIC für weitere Informationen.
12. Das Ausführen des Codes auf Ihrer FLEX Board hat das folgende Verhalten: das System auf der Platine blinkt mit einer Periode von 20 Sekunden und einer Einschaltdauer von ca. 6 Sekunden über 20. Die Erklärung ist die folgende:
    - Das System funktioniert wie eine synchrone Steuerung, mit einer Sampling-Frequenz von 0,1 Sekunden
    - Die *Sinus-Block-Ausgang* ist ein Sinus mit einer Frequenz von 0,05, was auf einen Zeitraum von 20 s entsprechen .
    - Der LED-Block ist direkt mit dem System verbunden und ist so programmiert das sich die LED einschaltet , wenn seine Eingabe größer als 0,5 ist.
    - Mit einem Blick auf die Abbildung unten, ist es klar, dass der Sinus-Wert größer als 0,5 für rund ein Drittel seiner Zeit ist. Da ist das System-LED für ca. 6 Sekunden über 20 Sekunden.

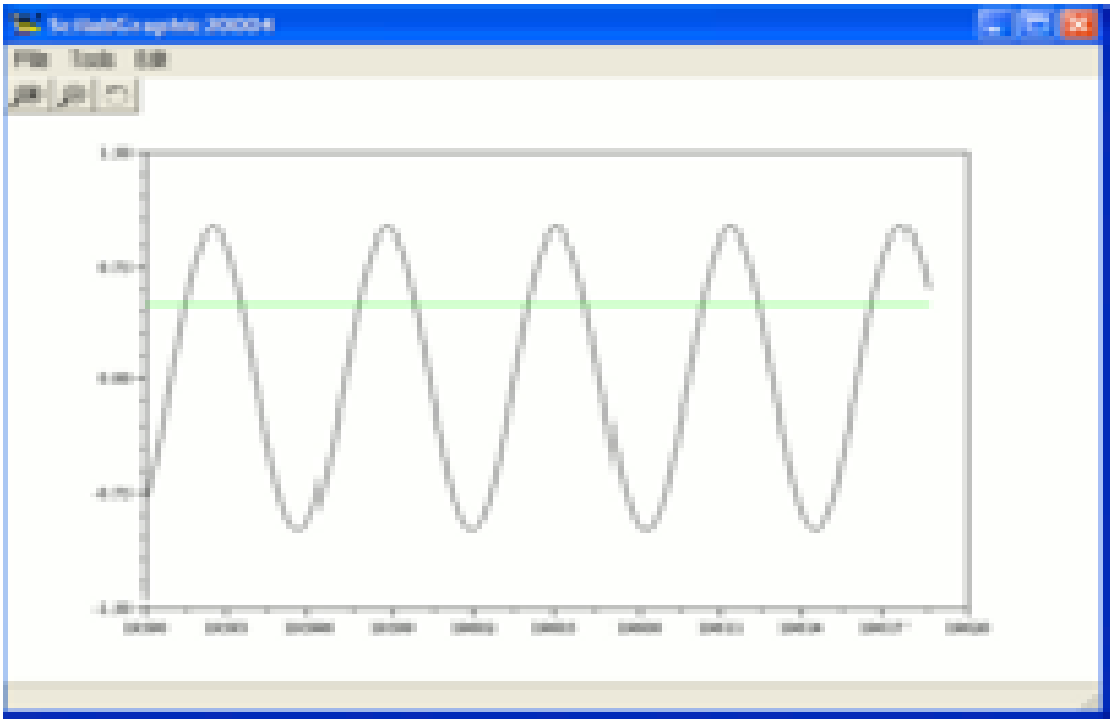


Abbildung 32: Eine Grafik einer Sinus-und einer konstanten wert 0,5

(Erika Enterprise, 2012)

5.Simulationsbeispiele

5.1 Aufbau und Simulation der Schaltungen

Dieser Abschnitt erklärt den Aufbau der Schaltungen und deren Simulationen .

Beispiel 1.

1. Starten Sie ScicosLab
2. Im Scicoslab-Menü Applications ⟹ Scicos auswählen
3. Wählen Sie Paletten aus dem Scicos-Menü
4. Im Scicoslab-Menü⇨Palette⇨Palettes⇨Sources und Sinks-Blöcke auswählen
5. Platzieren Sie die Blöcke und verbinden Sie diese

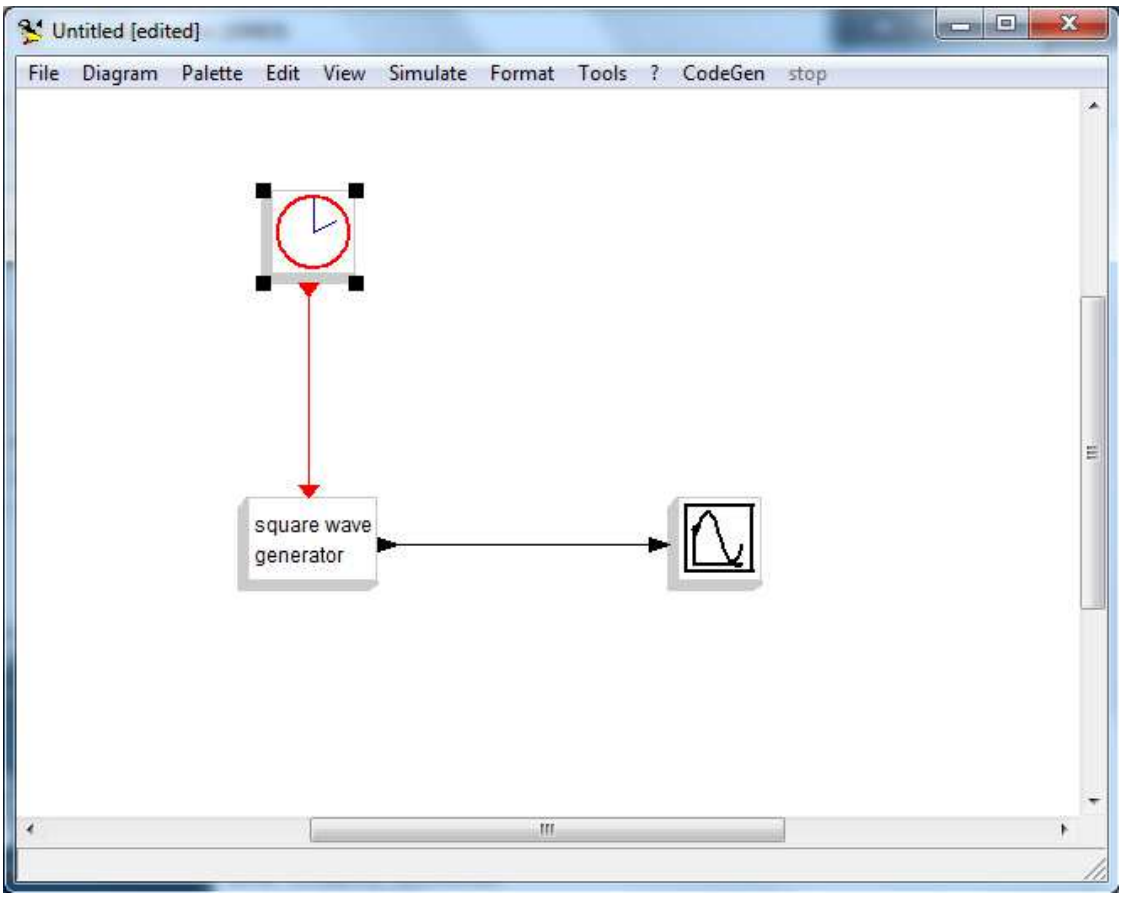


Abbildung 33: Aufbau der Schaltung

6. Stellen Sie Block-Parameter ein.

Der Clock Block Parameter

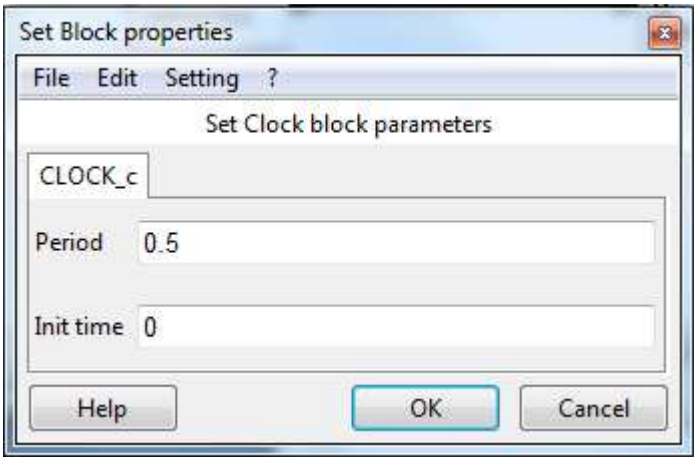


Abbildung 34: Clock Block Parameter

Erklärung zur Parametereinstellung

Periode: Auslösungshäufigkeit einer Messung. Je kleiner die Periode, desto mehr Punkte werden berechnet bzw. ausgelöst.

Initial time: Anfangszeit der ersten Auslösung einer Messung

Der Set Square Generator Block Parameter:

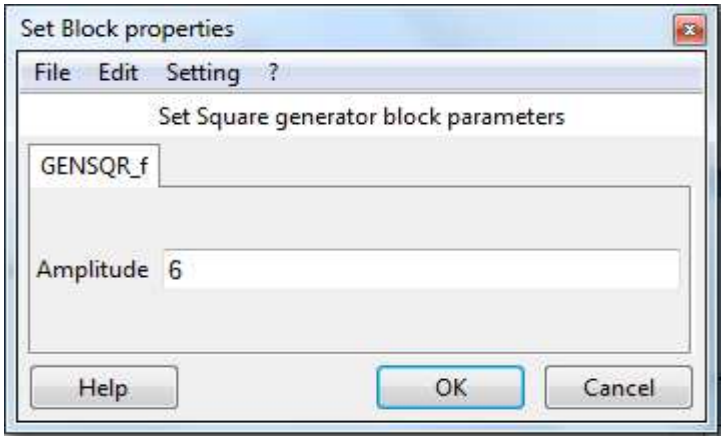


Abbildung 35: Square Generator Block Parameter

Der Set Scope Block Parameter:

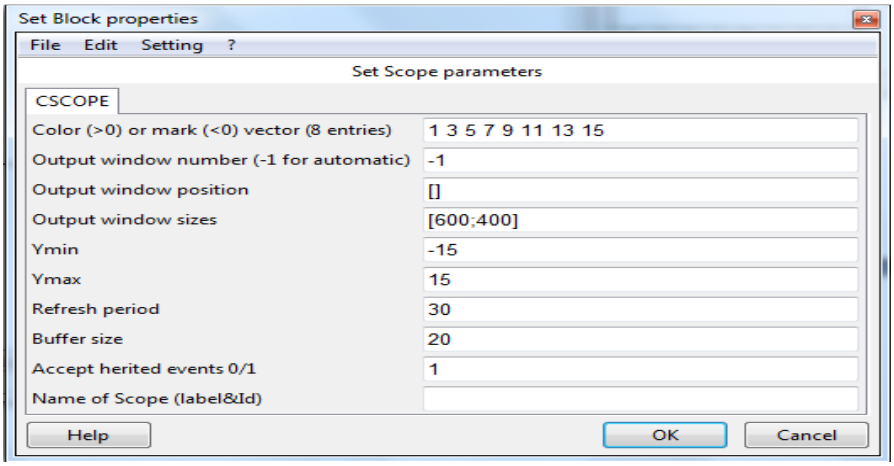


Abbildung 36: Set Scope Block Parameter

7. Simulieren Sie Das Modell.

   Im Scicoslab-Menü ⇨ Simulate ⇨ Run anklicken.

   Es erscheint die folgende Grafik Die folgende Abbildung zeigt die Simulationsergebnisse

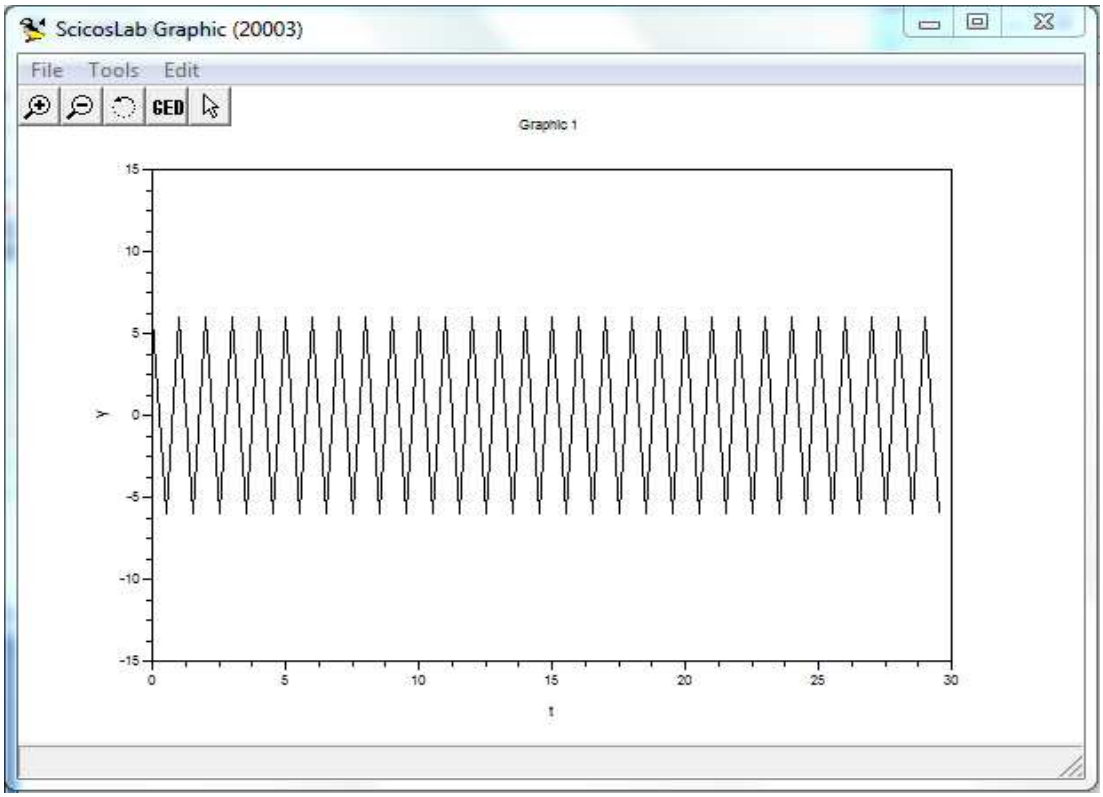


Abbildung 37: Messauslösung mit Perioden-Wert 0.5

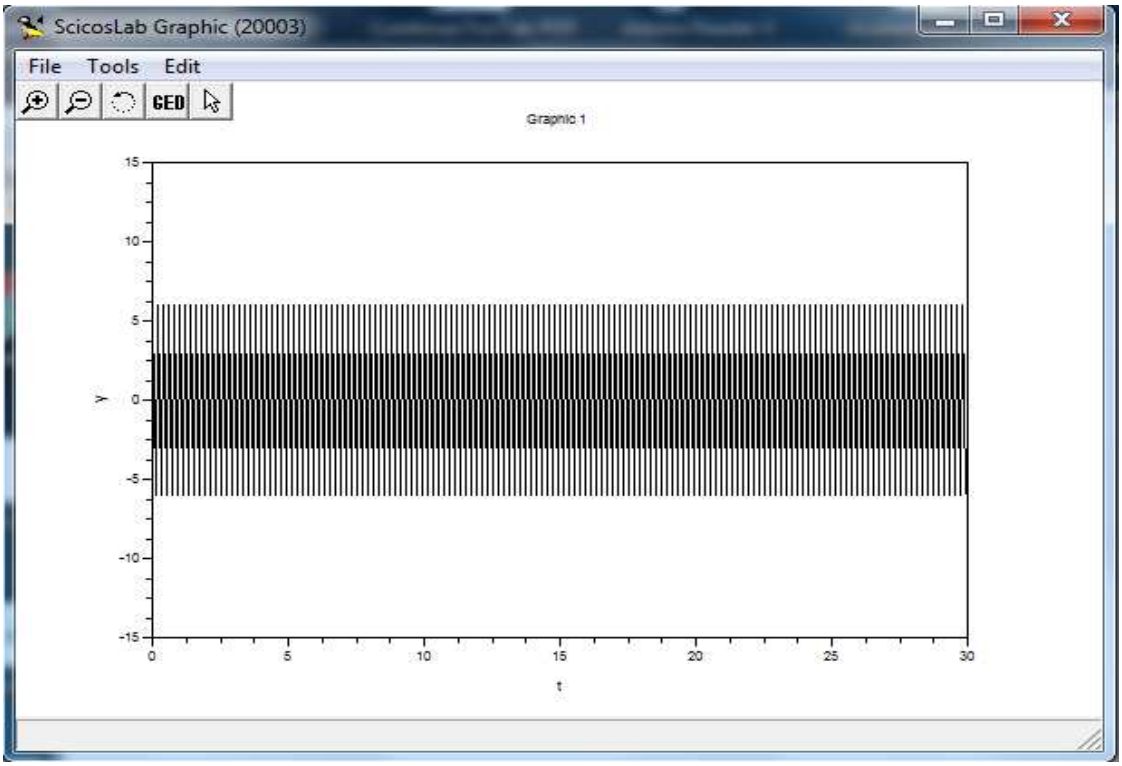


Abbildung 38 : Messauslösung mit Perioden-Wert 0.1

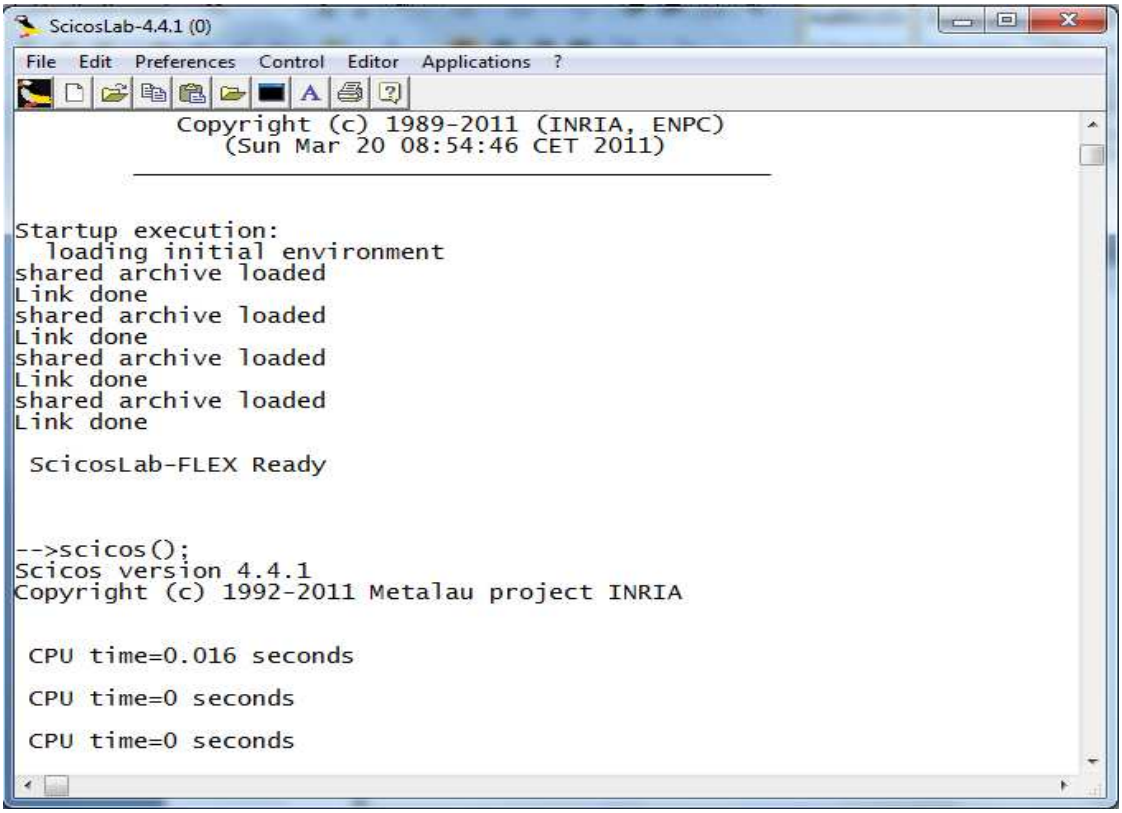


Abbildung 39: Scicoslab - Oberfläche

Beispiel 2:

1. Starten Sie ScicosLab
2. Im Scicoslab-Menü Applications ⟹ Scicos auswählen
3. Wählen Sie Paletten aus dem Scicos-Menü
4. Im Scicoslab-Menü⇨Palette⇨Palettes⇨Sources und Sinks-Blöcke auswählen
5. Platzieren Sie die Blöcke und verbinden Sie es

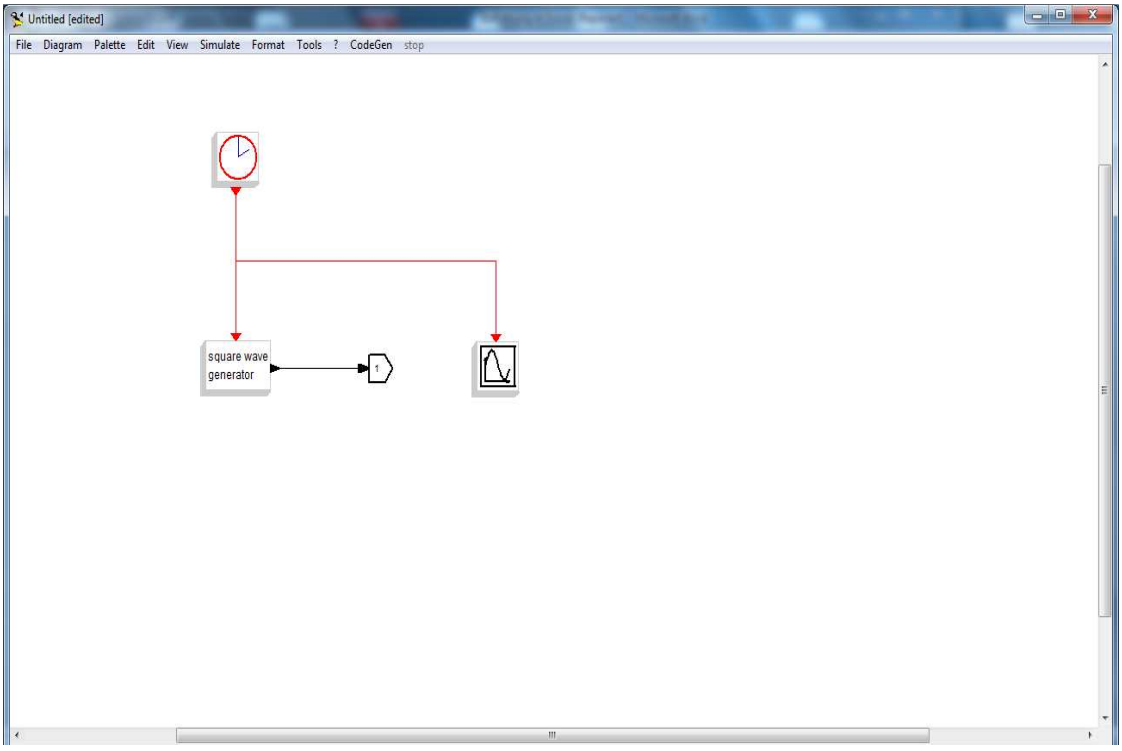


Abbildung 40: Aufbau der Schaltung

6. Stellen Sie Block-Parameter ein.

   Der Clock Block Parameter:

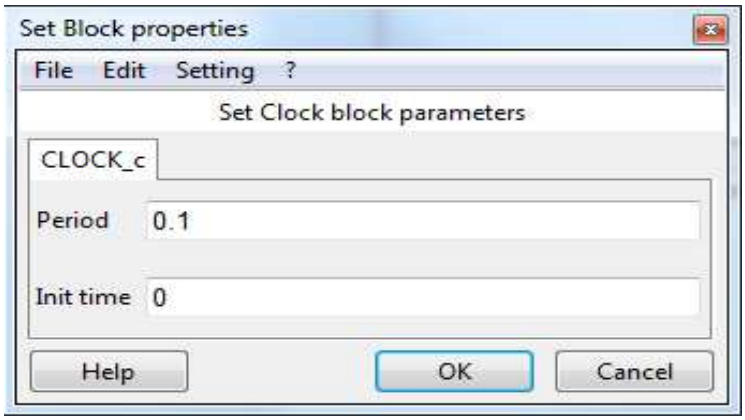


Abbildung 41: Clock Block Parameter

Der Set Square Genrator BlockParameter:

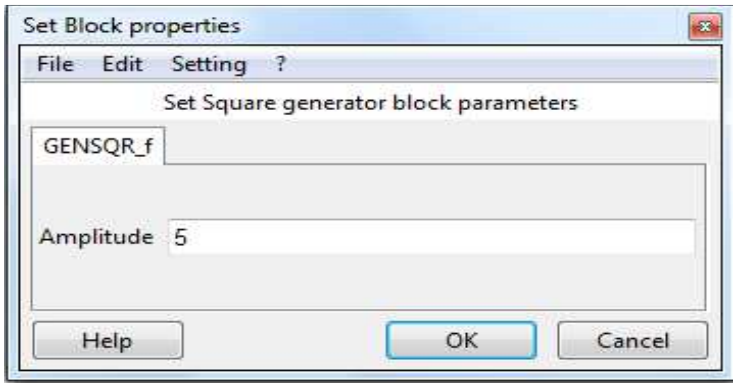


Abbildung 42 : Set Block Parameter

Der Set Scope Block Parameter:

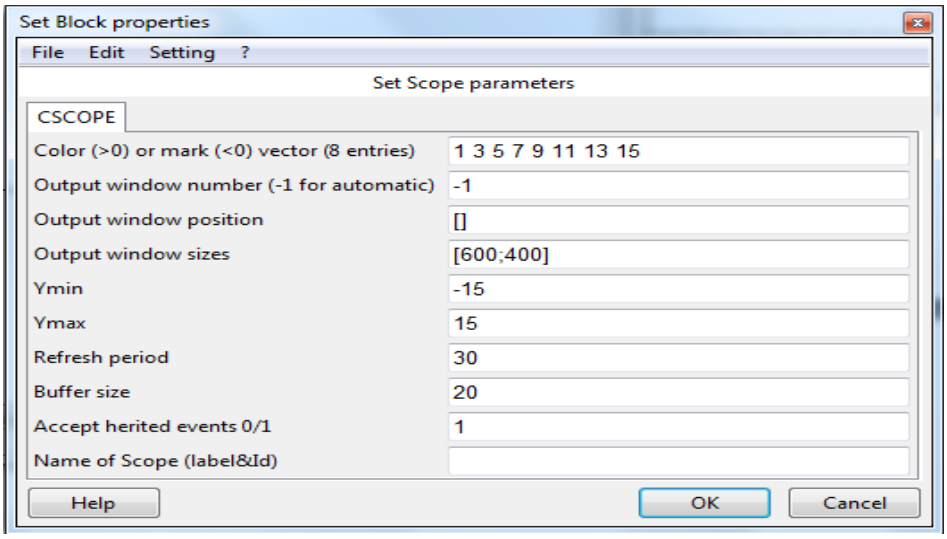


Abbildung 43 : Set Scope Block Parameter

7. Simulieren Sie Das Modell.

   Im Scicoslab-Menü ⇨ Simulate ⇨ Run anklicken. Es erscheint die folgende Grafik.

Die folgende Abbildung zeigt die Simulationsergebnisse

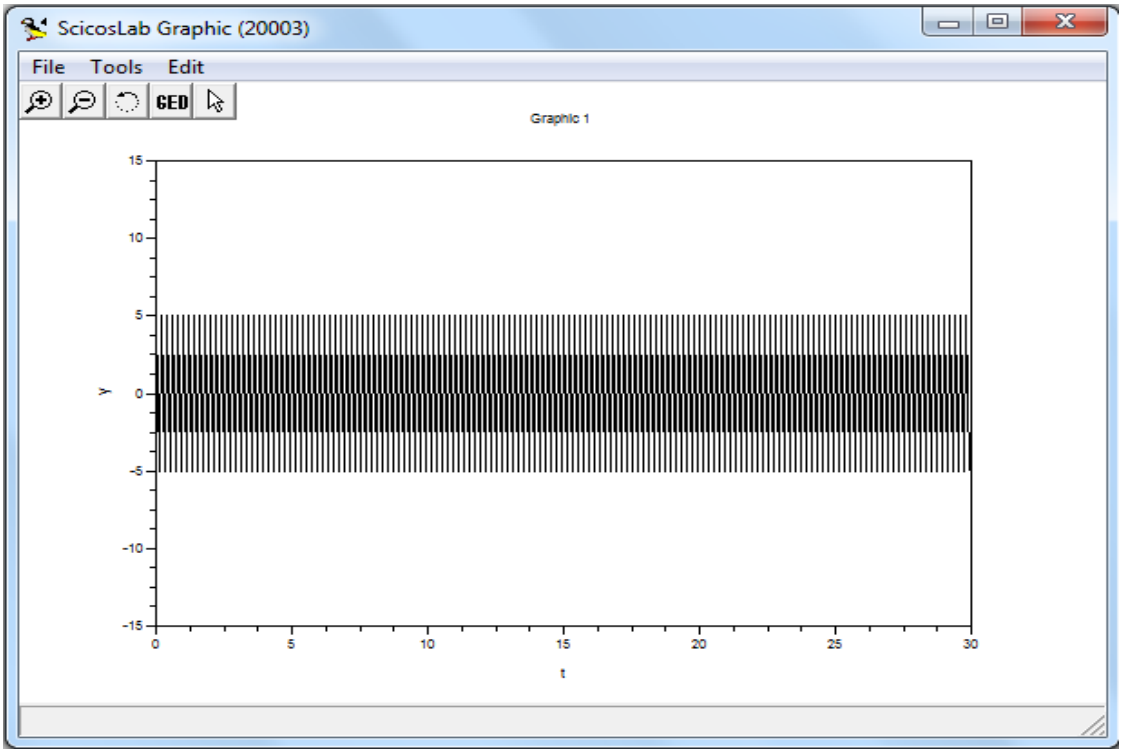


Abbildung 44 : Simulation der Schaltung

Beispiel 3: DC-Motor

1. Starten Sie ScicosLab.
2. Im Scicoslab-Menü Applications ⟹ Scicos auswählen .
3. Wählen Sie Paletten aus dem Paletten-Menü
4. Öffnen Sie File ⇨ Open ⇨ Scicos_ee ⇨ Exemples ⇨ Scicos_flex ⇨ DC Motor ⇨ DC Motor-simulation . Wie gezeigt in Abbildung

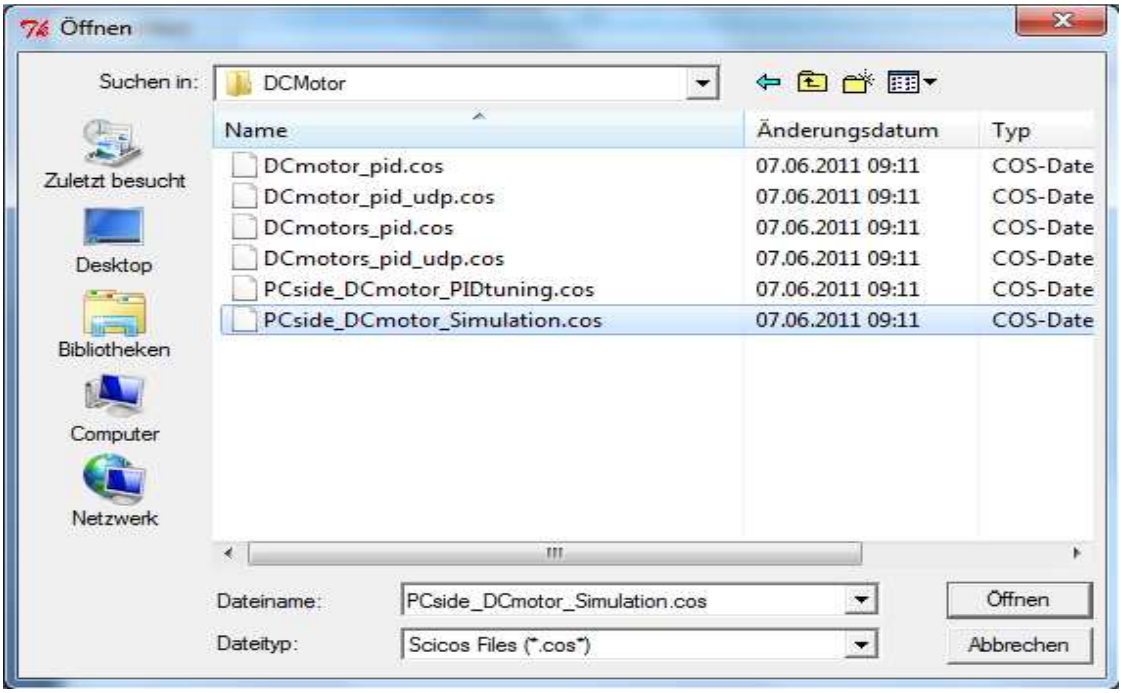


Abbildung 45 : DC Motor Beispiel

5. Ein Fenster erscheint, mit einigen Blöcke für DC-Motor Super Block, wie gezeigt in Abbildung

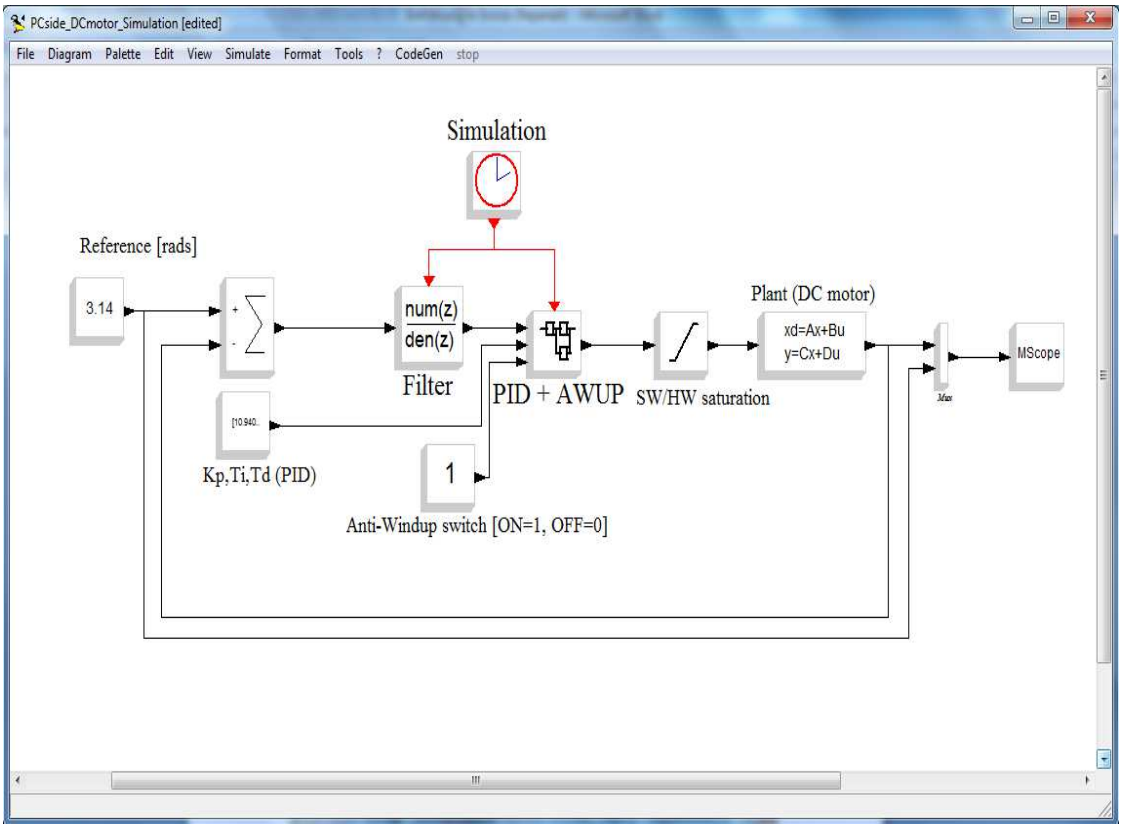


Abbildung 46: DC-Motorschaltbild

6. Als Ergebnis ist ein Scicoslab Graphik. (siehe Abbildung)

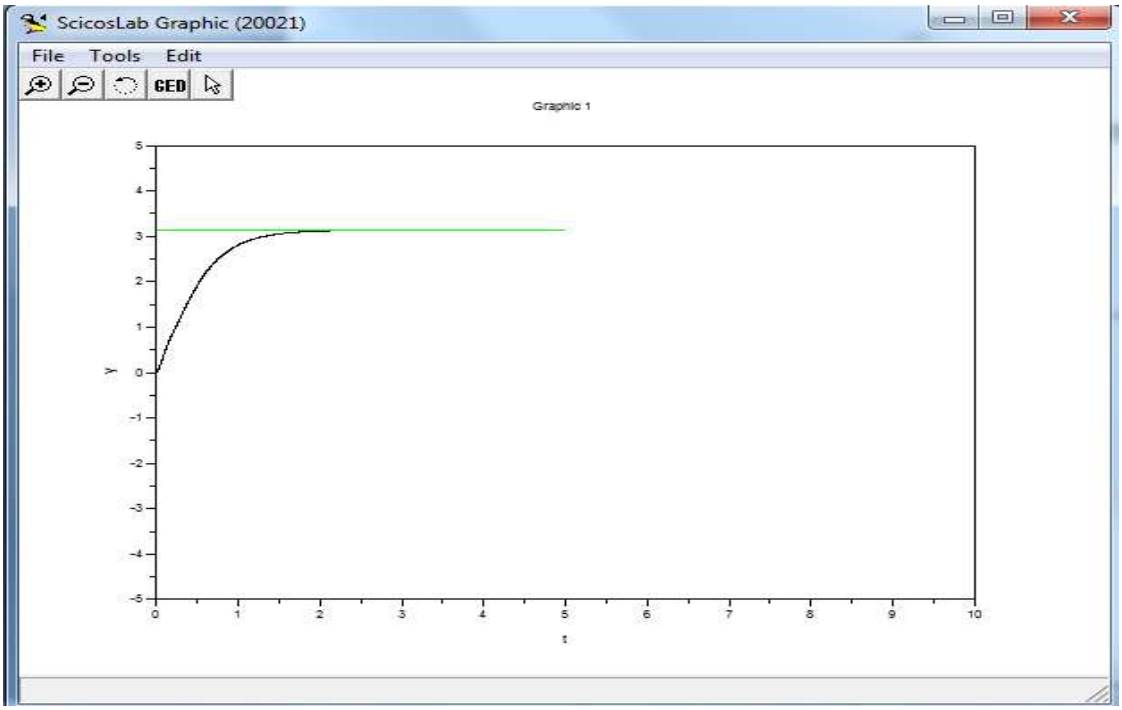


Abbildung 47: Simulationsergebnisse

Wenn Sie den PID+AWUP- Block im DC-Motor Schaltbild anklicken, zeigt sich das äußere Super Block Schaltbild. (Siehe Abbildung)

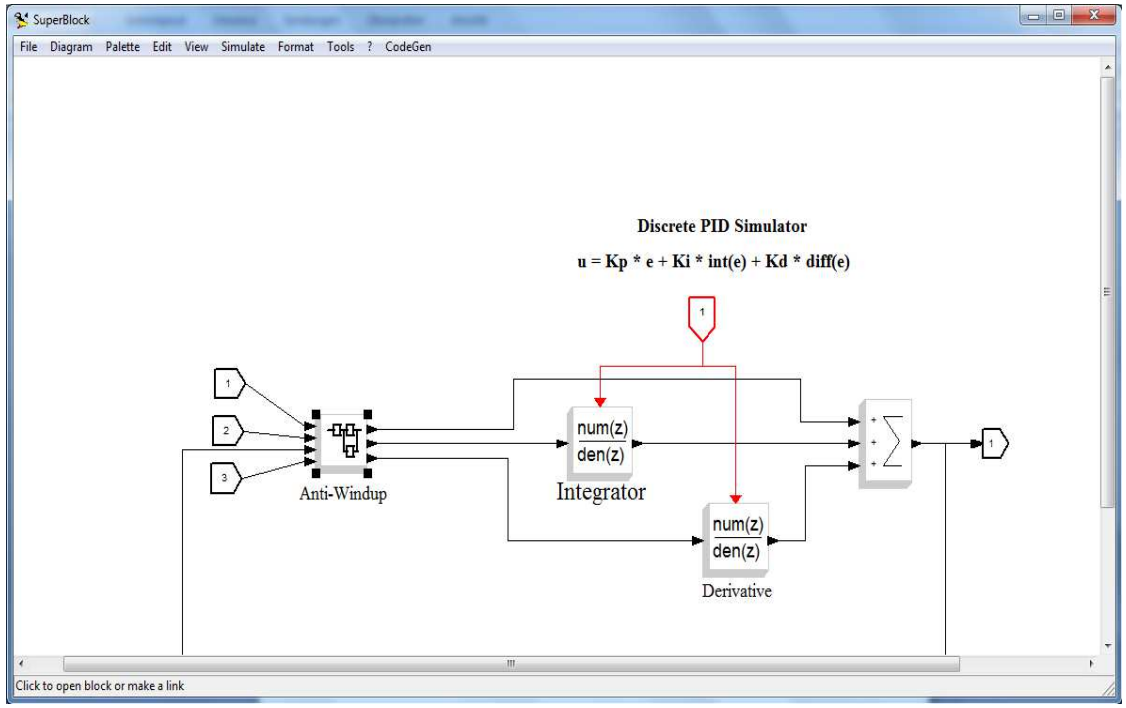


Abbildung 48: äußeres Superblock Schaltbild

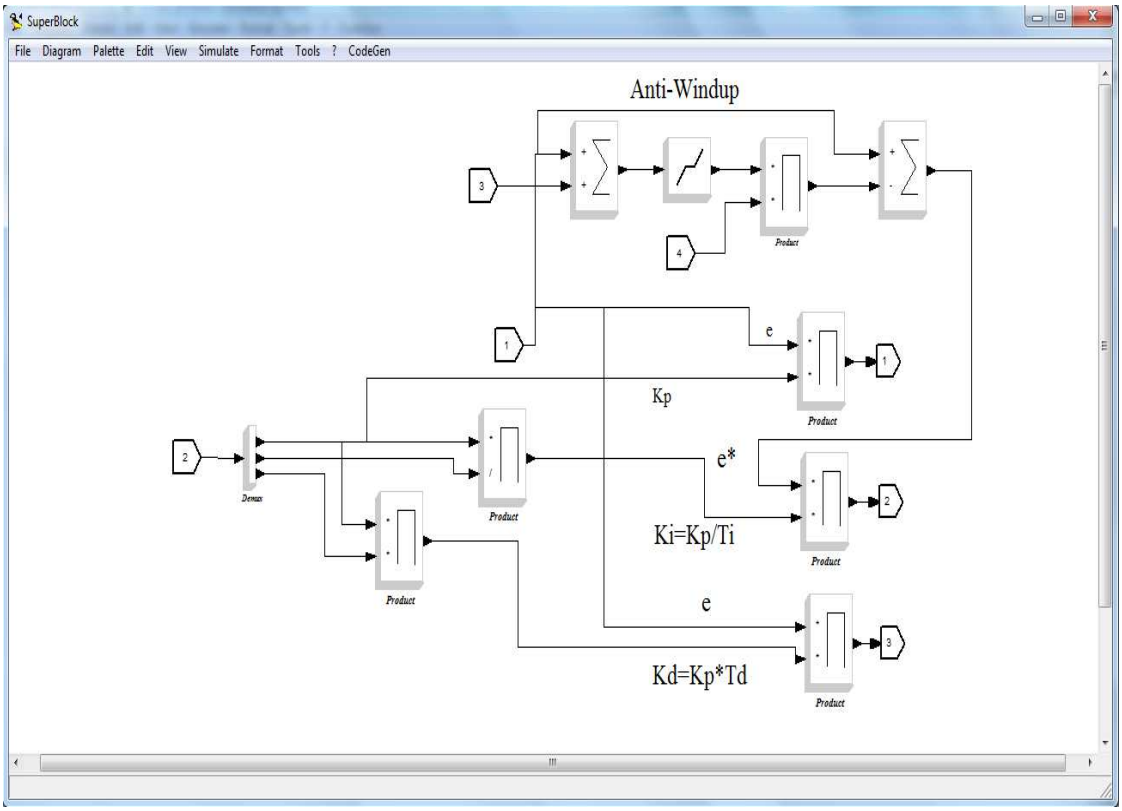


Abbildung 49: Demo Board Flex-Side -Simulation

Nach der Einstellung des Anti-Windup Parameter, wie gezeigt in Abbildung

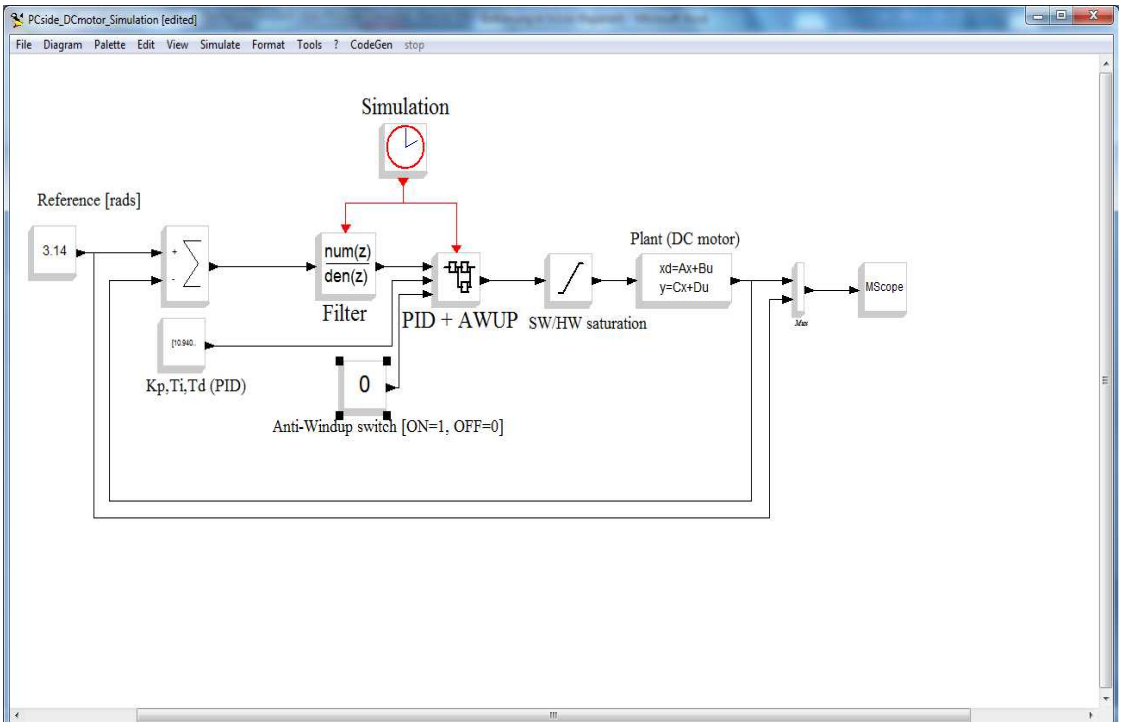


Abbildung 50: Anti-Windup Parameter mit dem Wert 0

Das Ergebnis ist eine Scicoslab Graphik. (siehe Abbildung)

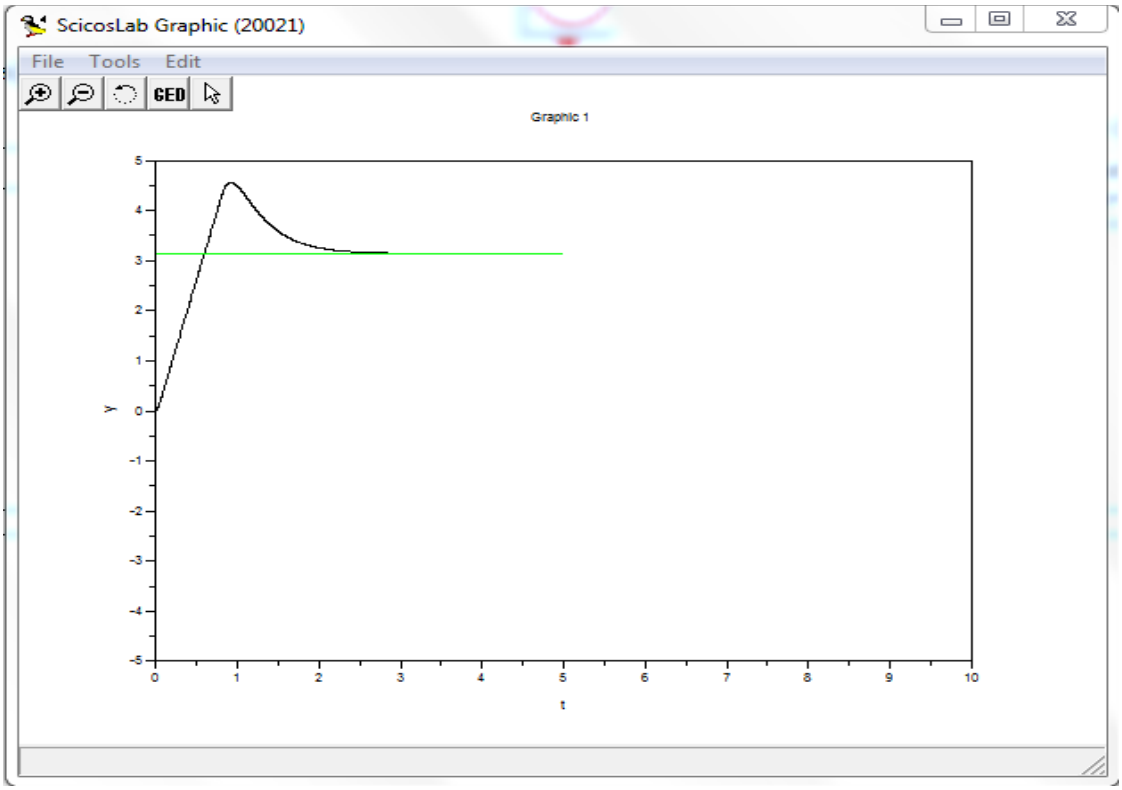


Abbildung 51 : Scicoslab Graphik

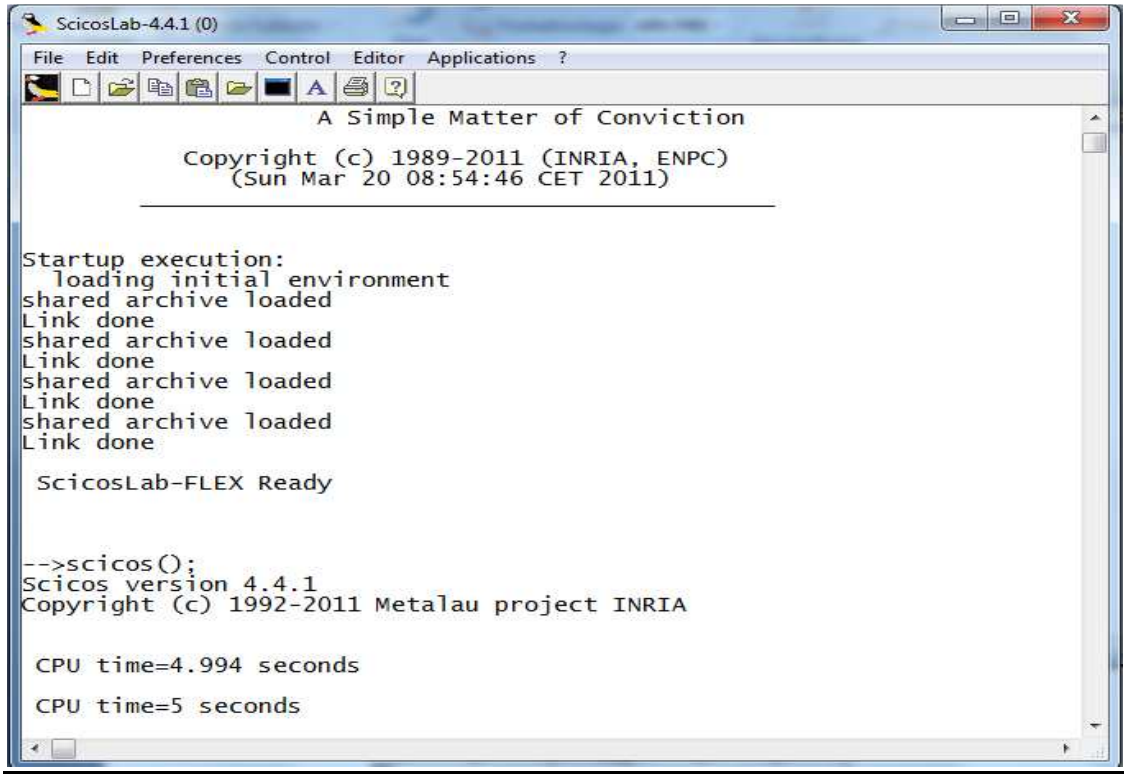


Abbildung 52 : Scicoslab-Oberfläche

Beispiel 4:

Demo Board FlexBlock -Schaltbild

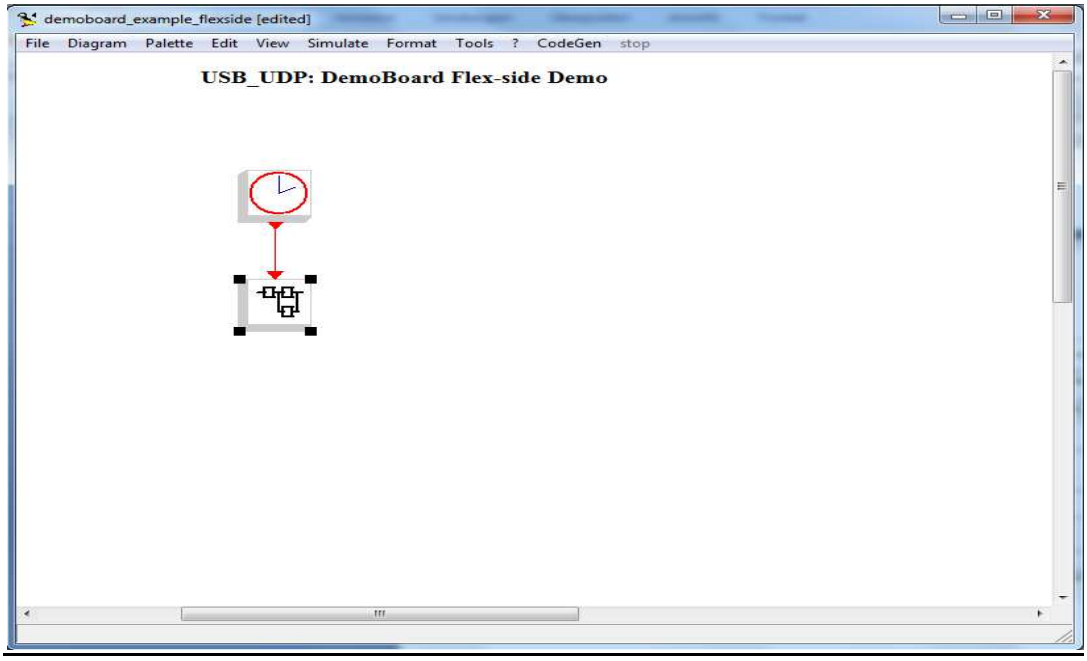


Abbildung 53 : Demo Board-Flexside Schaltbild

Super Block aus Demo Board FlexBlockschaltbild

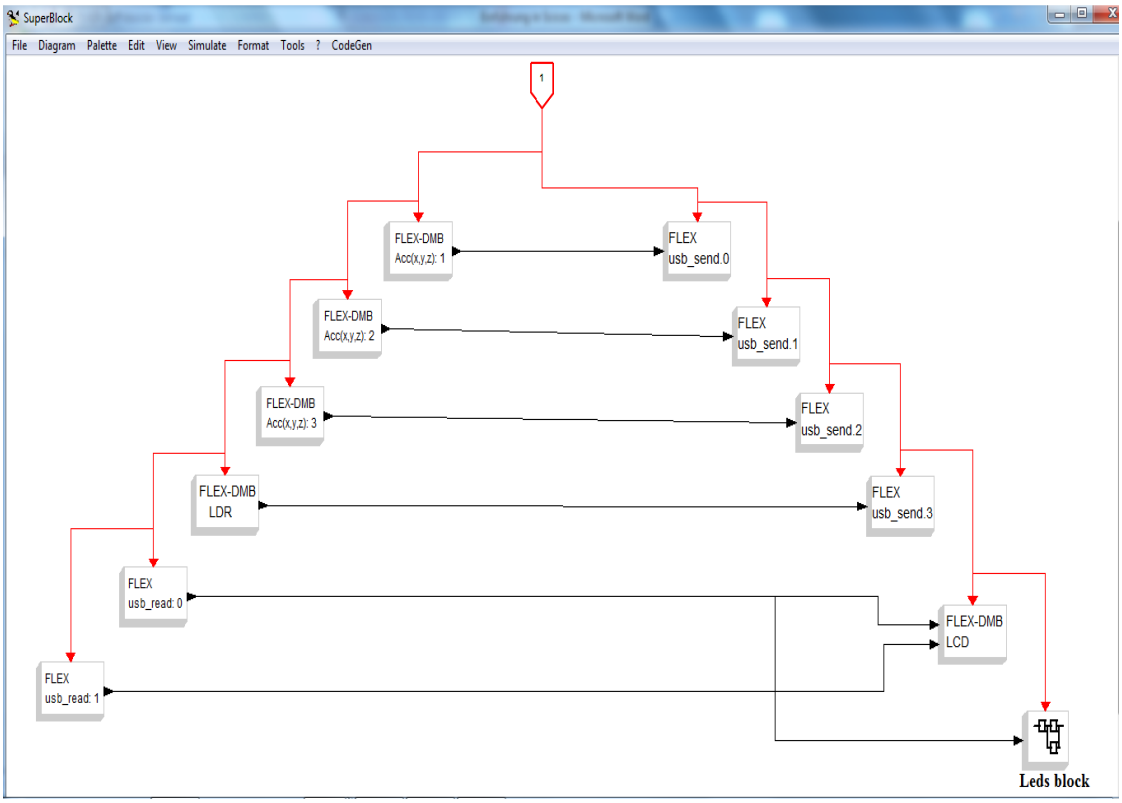


Abbildung 54: Demo Board-Flexside-Super Block Schaltbild

6.Handbuch zur Handhabung der Simulationsumgebung

(Siehe Abschnitt 4 und 5)

Literaturverzeichnis

INRIA, Paris-Rocquencourt center. (2012). http://www.scicos.org/.

Barucki, D.-I. T. (kein Datum). http://swt.cs.tu-berlin.de/asim-sts-05/tagband/barucki.pdf. Abgerufen am 2012

Erika Enterprise. (2012). http://erika.tuxfamily.org/flex.html.

http://help.sap.com/printdocu/core/Print46c/de/data/pdf/PPREM/PPREM.pdf. (kein Datum). Abgerufen am 2012

http://regpro.mechatronik.uni-linz.ac.at/downloads/autvertpr/Aufgabe%201.pdf. (kein Datum). Abgerufen am 2011

Kepler, J. (kein Datum). http://regpro.mechatronik.uni-linz.ac.at/downloads/autvertpr/Aufgabe%201.pdf. Abgerufen am 11 2011

Microchip. (kein Datum). http://www.microchip.com. Abgerufen am 2011

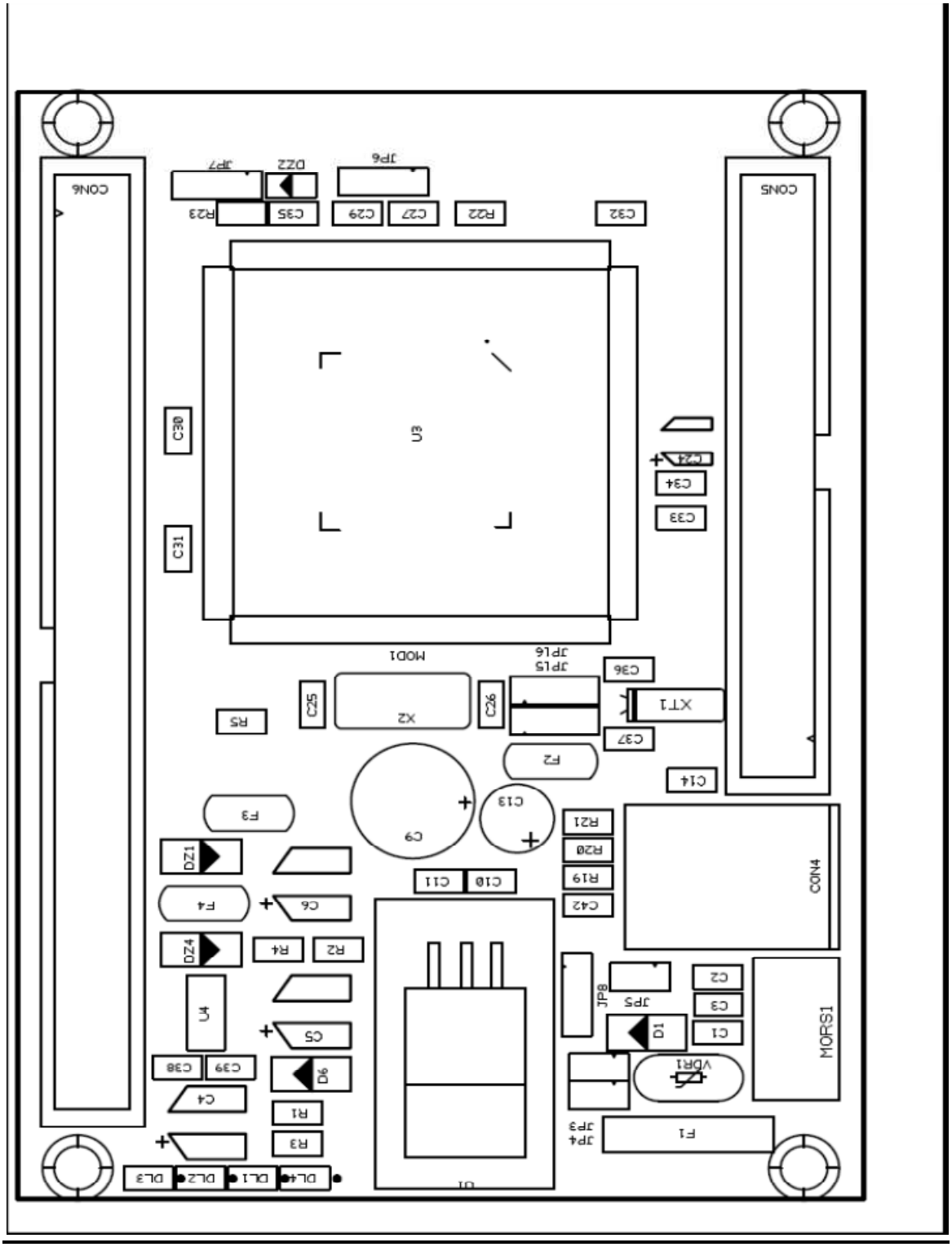
N.Bajcinca. (kein Datum). http://www.xpclabdesk.com/doc/AT04_Bajcinca_1.pdf. Abgerufen am 11 2011

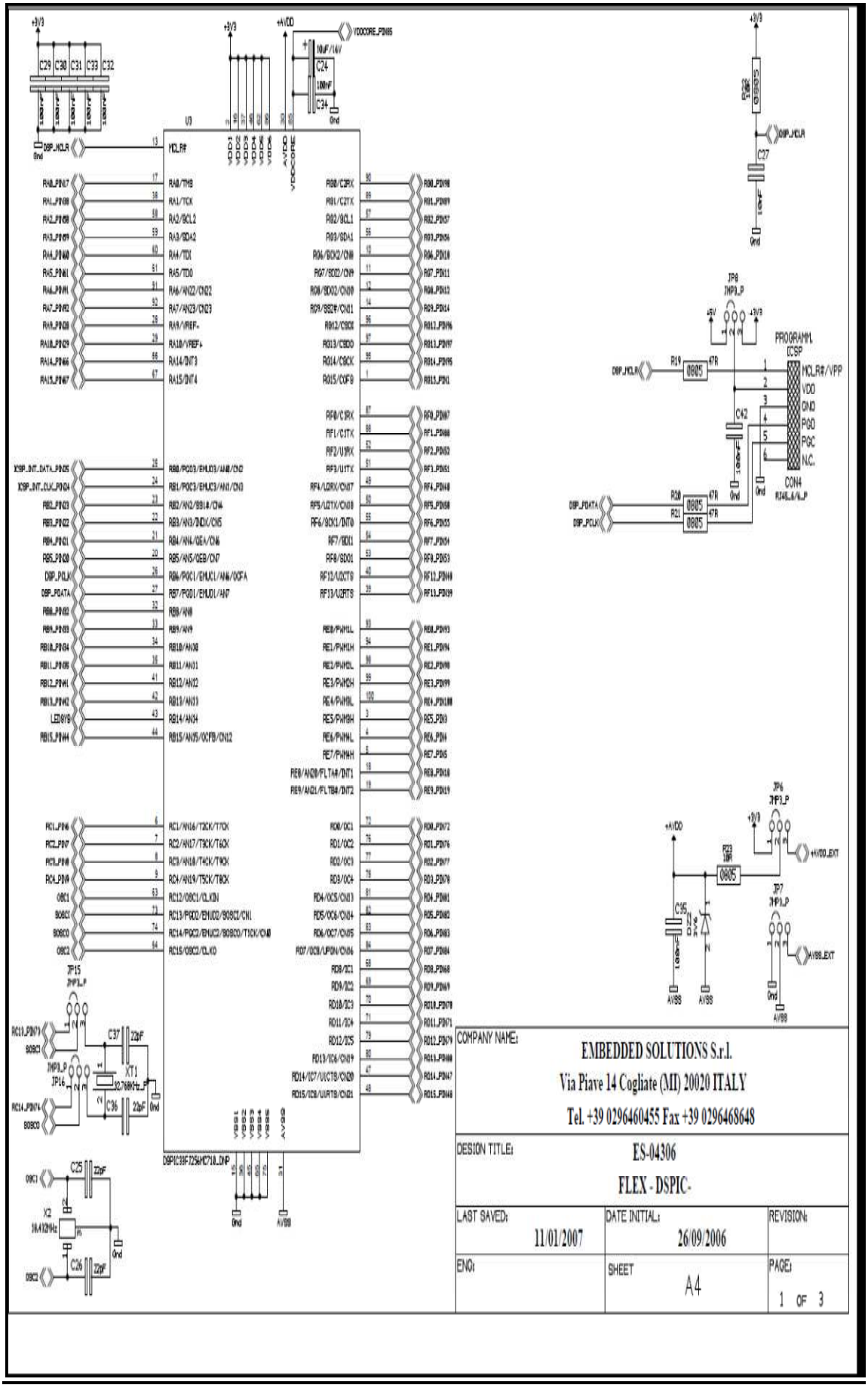
Nikoukhah, R. (kein Datum). http://www.seit.adfa.edu.au/staff/sites/hrp/teaching/ScicosNikoukhah.pdf. Abgerufen am 2011

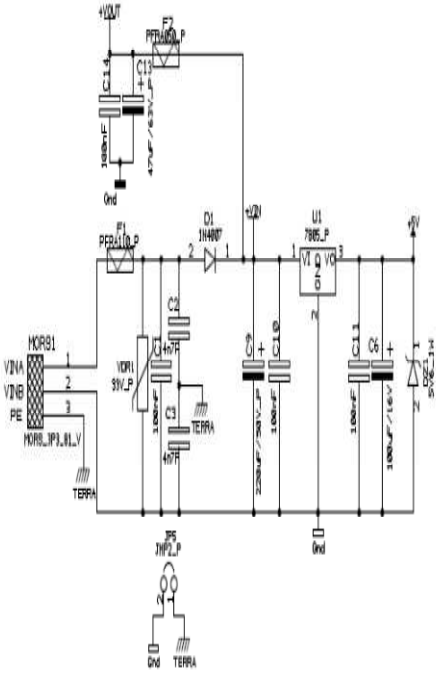
SAP AG. (kein Datum). http://help.sap.com/printdocu/core/Print46c/de/data/pdf/PPREM/PPREM.pdf. Abgerufen am 2011

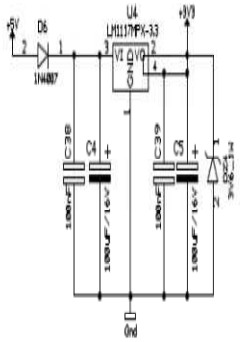
Anhang

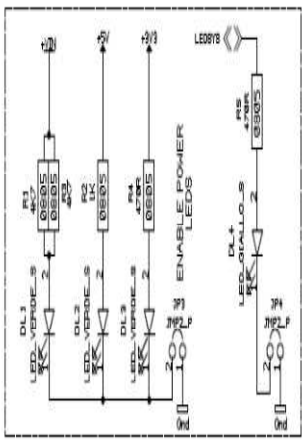
Flex-light Board-Schemas



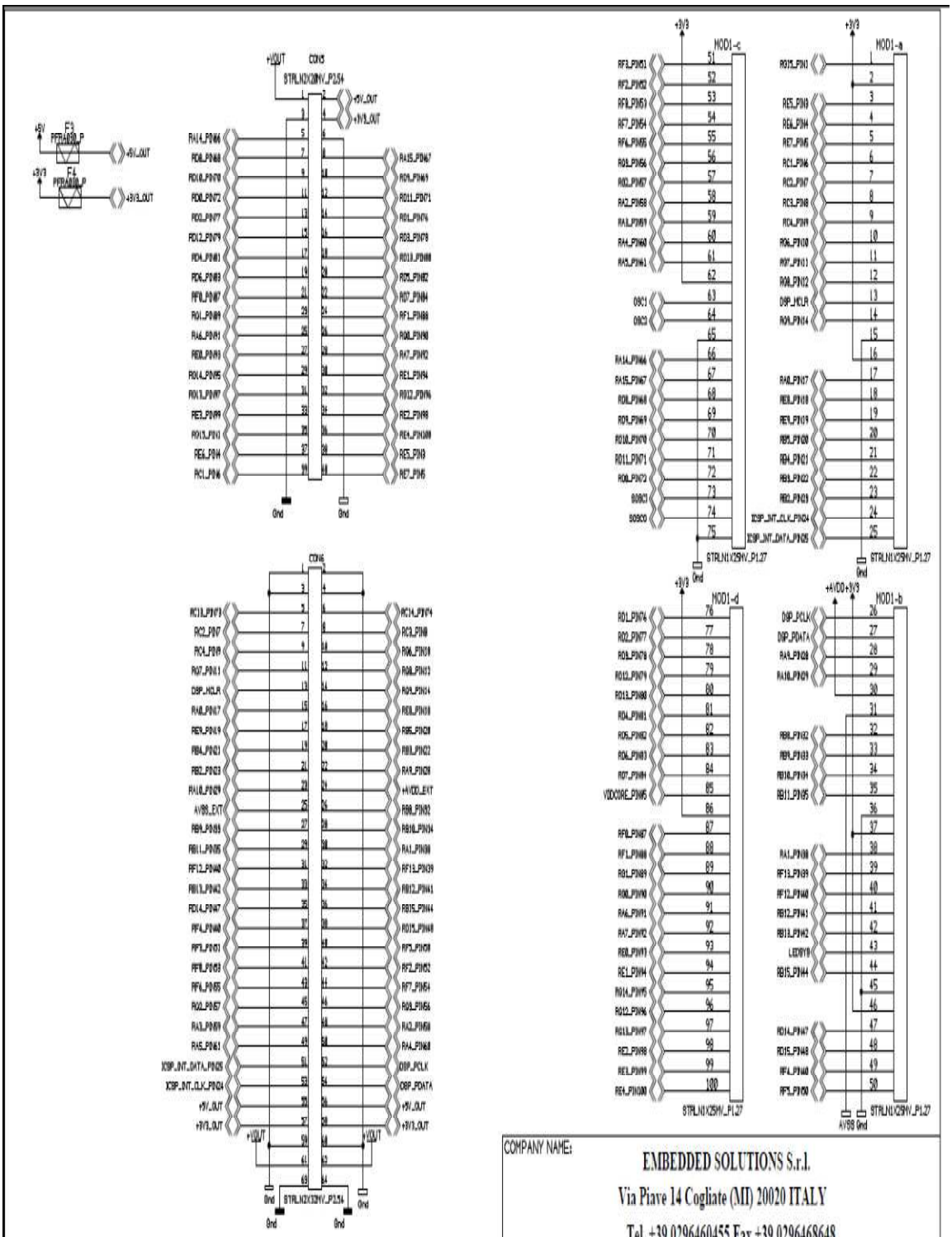

COMPANY NAME:
EMBEDDED SOLUTIONS S.r.l.
Via Piave 14 Cogliate (MI) 20020 ITALY
Tel. +39 0296460455 Fax +39 0296468648
DESIGN TITLE:
ES-04306
FLEX - DSPIC-
LAST SAVED:
11/01/2007
DATE INITIAL:
26/09/2006
REVISION:
ENG:
SHEET
A4
PAGE:
1 OF 3

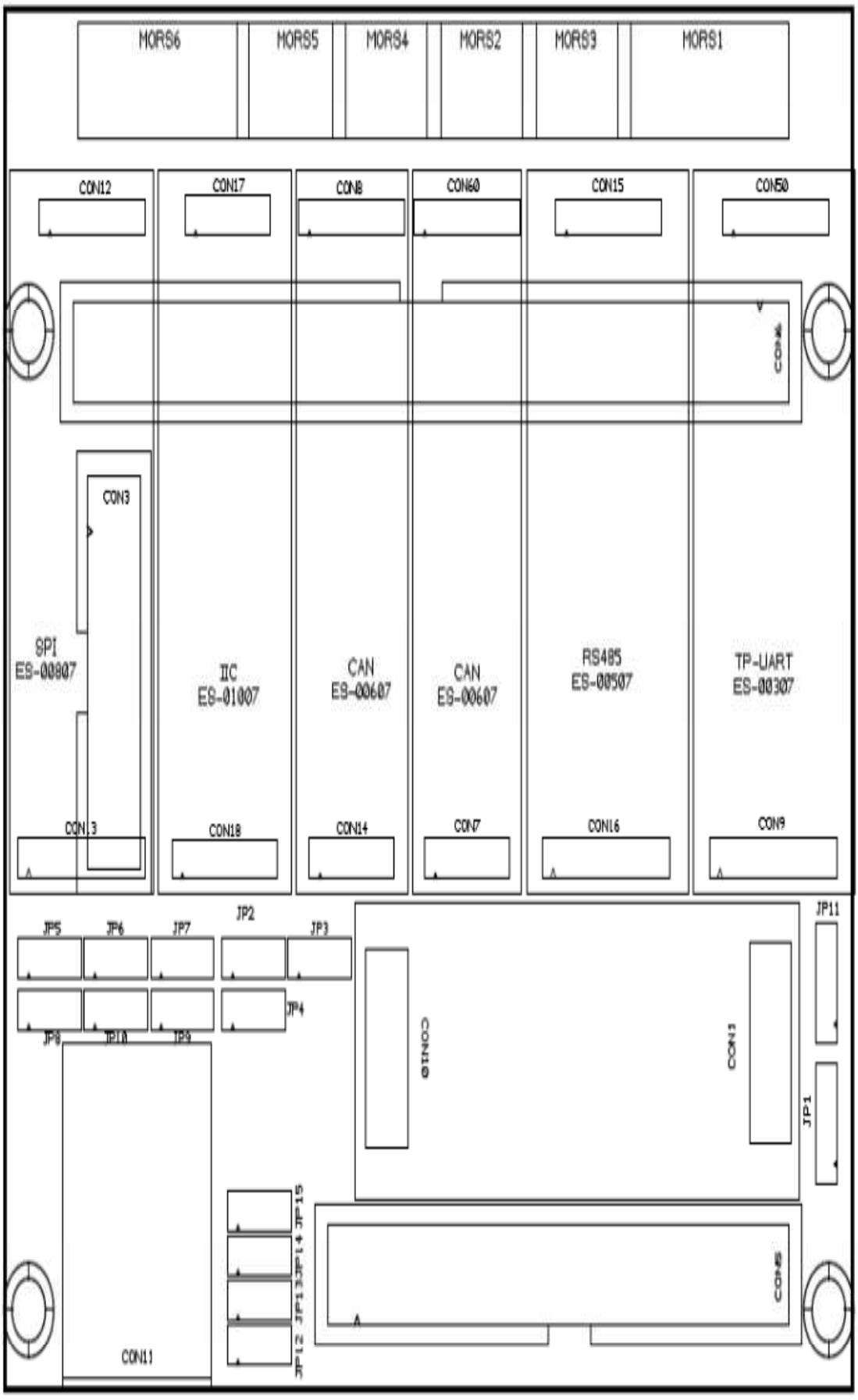


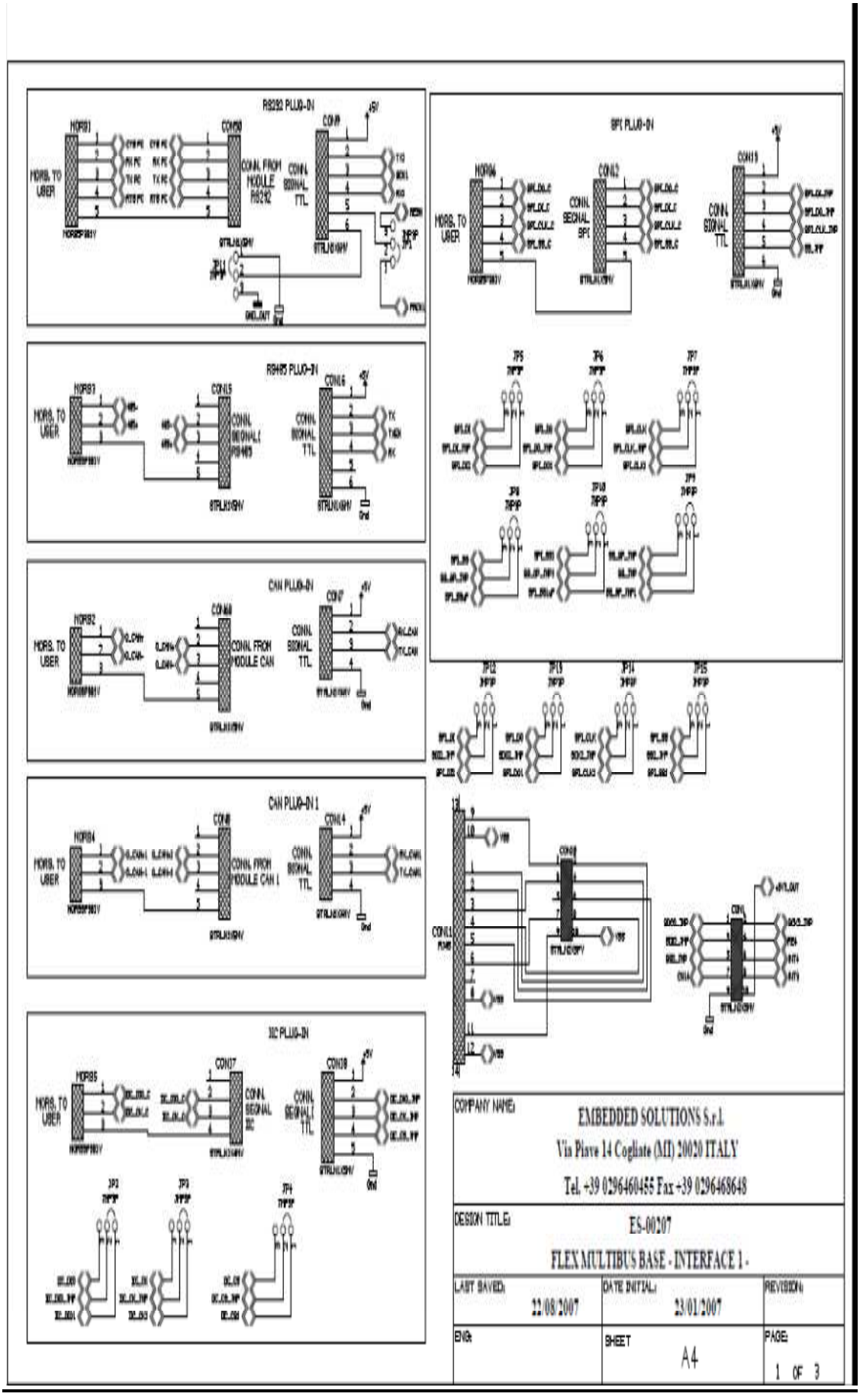


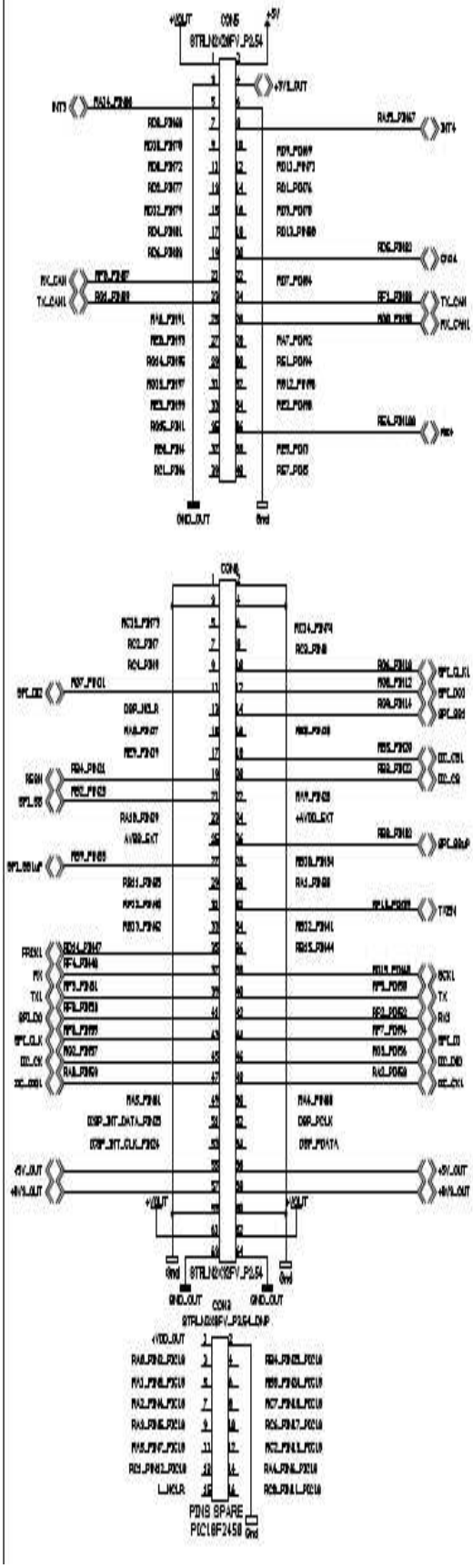


COMPANY NAME: EMBEDDED SOLUTIONS S.r.l. Via Piave 14 Cogliate (MI) 20020 ITALY Tel. +39 0296460455 Fax +39 0296468648		
DESIGN TITLE: ES-04306 FLEX - POWER -		
LAST SAVED: 11/01/2007	DATE INITIAL: 26/09/2006	REVISION:
ENG:	SHEET A4	PAGE: 2 OF 3


COMPANY NAME:
EMBEDDED SOLUTIONS S.r.l.
Via Piave 14 Cogliate (MI) 20020 ITALY
Tel. +39 0296460455 Fax +39 0296468648
DESIGN TITLE:
ES-04306
FLEX - DAUGHTER CONNECTORS -
LAST SAVED:
11/01/2007
DATE INITIAL:
26/09/2006
REVISION:
ENG:
SHEET
A4
PAGE:
3 OF 3


MORS6
MORS5
MORS4
MORS2
MORS3
MORS1
CON12
CON17
CON8
CON60
CON15
CON50
CON5
CON3
SPI
ES-00807
IIC
ES-01007
CAN
ES-00607
CAN
ES-00607
RS485
ES-00507
TP-UART
ES-00307
CON13
CON18
CON14
CON7
CON16
CON9
JP2
JP5
JP6
JP7
JP3
JP4
JP8
JP10
JP9
JP11
CON10
CON3
JP1
JP15
JP14
JP13
JP12
CON11
CON6

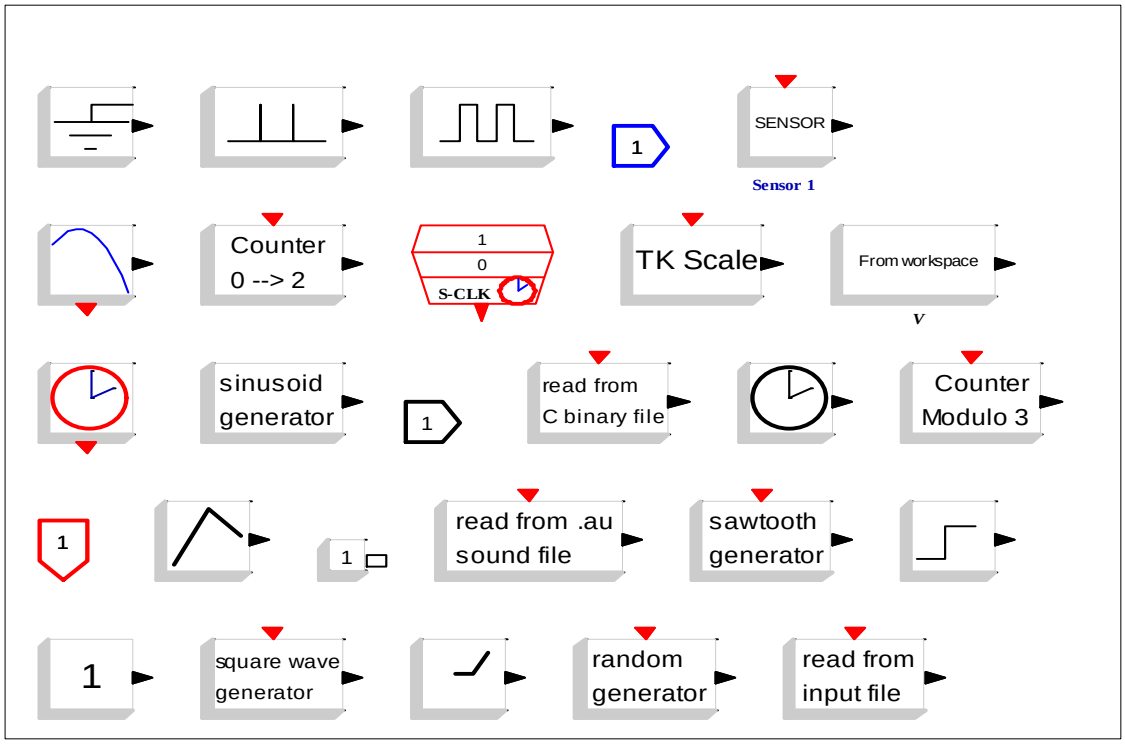

RS232 PLUG-IN
MORS. TO USER
CONN. SIGNAL TTL
RS485 PLUG-IN
CAN PLUG-IN
CAN PLUG-IN 1
I2C PLUG-IN
SPI PLUG-IN
COMPANY NAME:
EMBEDDED SOLUTIONS S.r.l.
Via Piave 14 Cogliate (MI) 20020 ITALY
Tel. +39 0296460455 Fax +39 0296468648
DESIGN TITLE:
ES-00207
FLEX MULTIBUS BASE - INTERFACE 1 -
LAST SAVED:
22/08/2007
DATE INITIAL:
23/01/2007
REVISION:
ENG:
SHEET
A4
PAGE:
1 OF 3



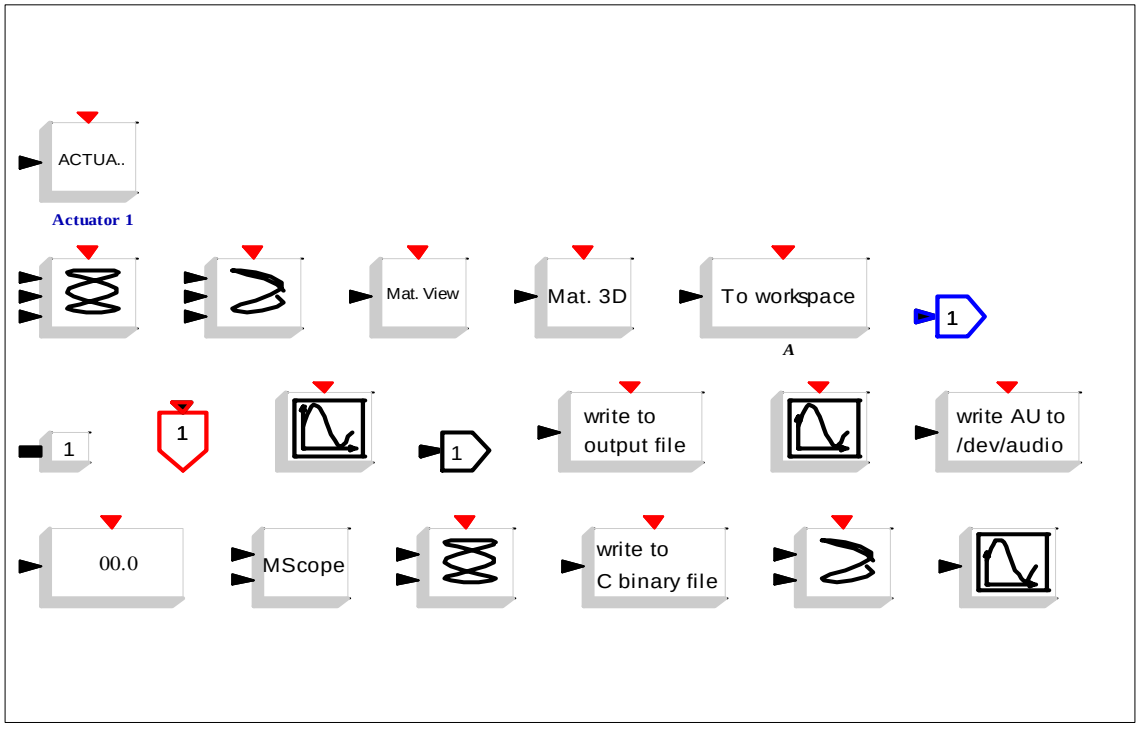
COMPANY NAME:	EMBEDDED SOLUTIONS S.r.l. Via Piave 14 Cogliate (MI) 20020 ITALY Tel. +39 0296460455 Fax +39 0296468648	
DESIGN TITLE:	ES-00207 FLEX MULTIBUS BASE - INTERFACE 2 -	
LAST SAVED: 22/08/2007	DATE INITIAL: 23/01/2007	REVISION:
ENG:	SHEET A4	PAGE: 2 OF 3

Die einzelnen Paletten

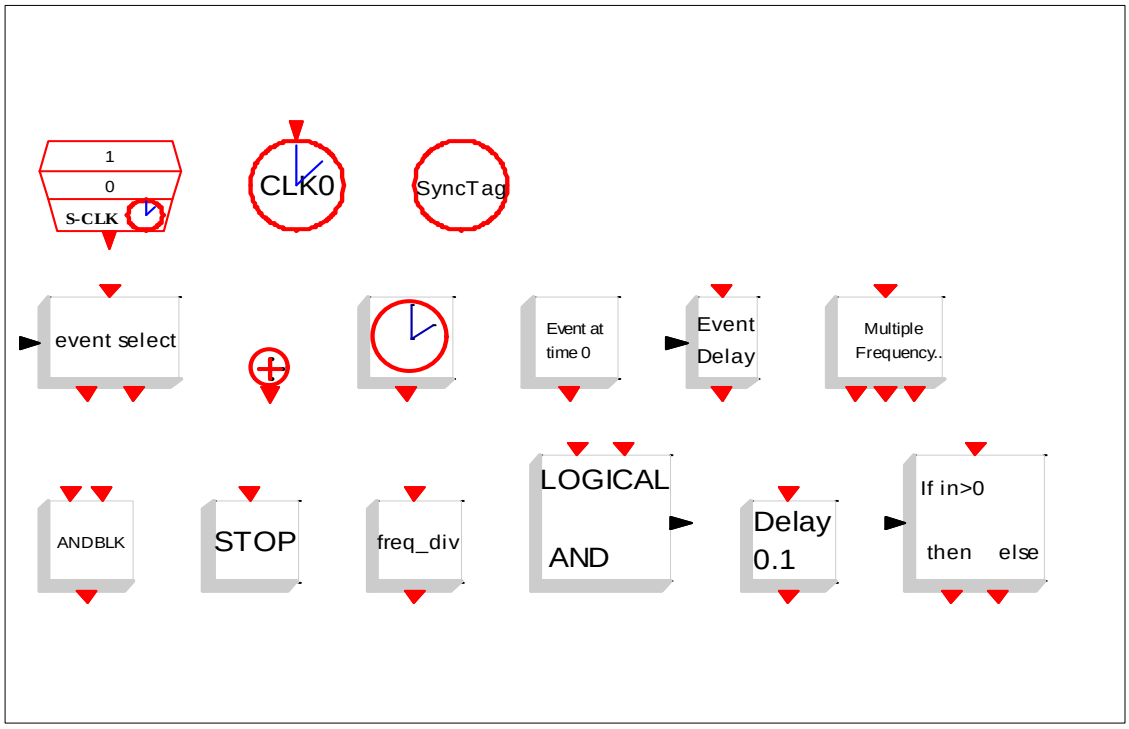
Die Sources-Palette



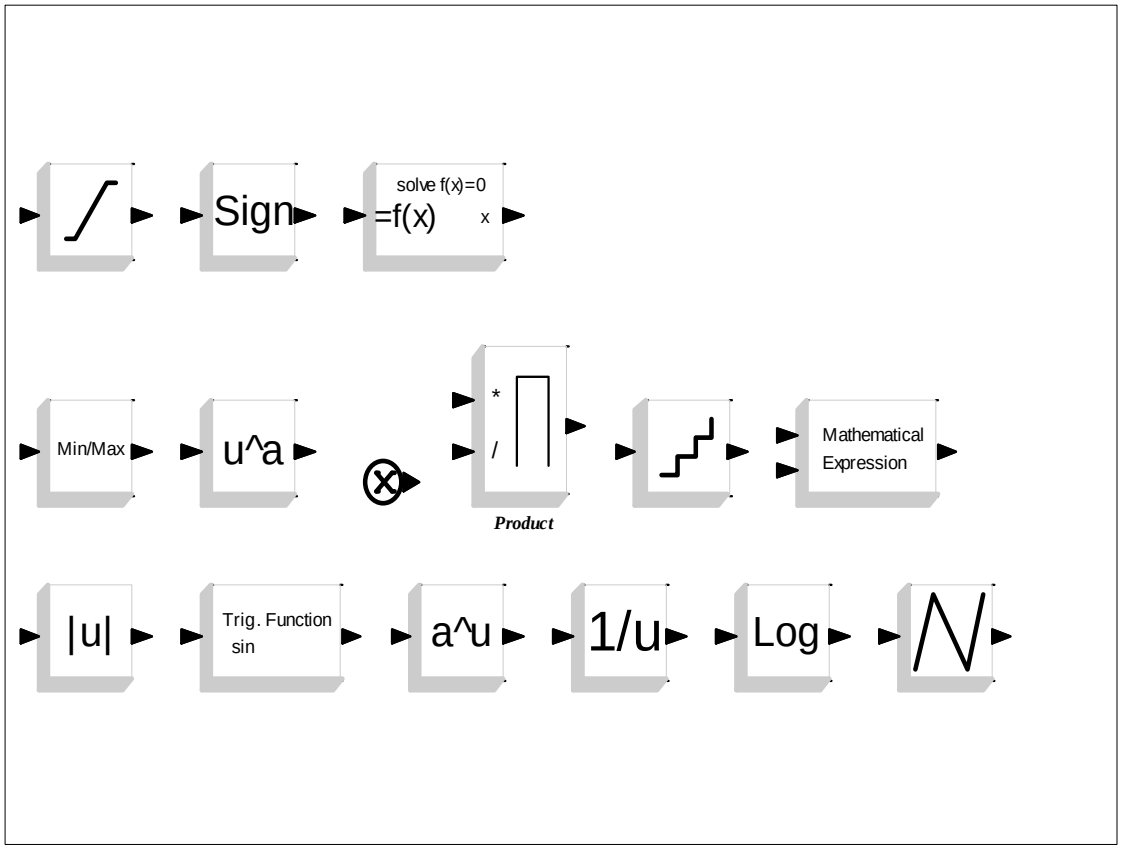
Die Sinks-Palette



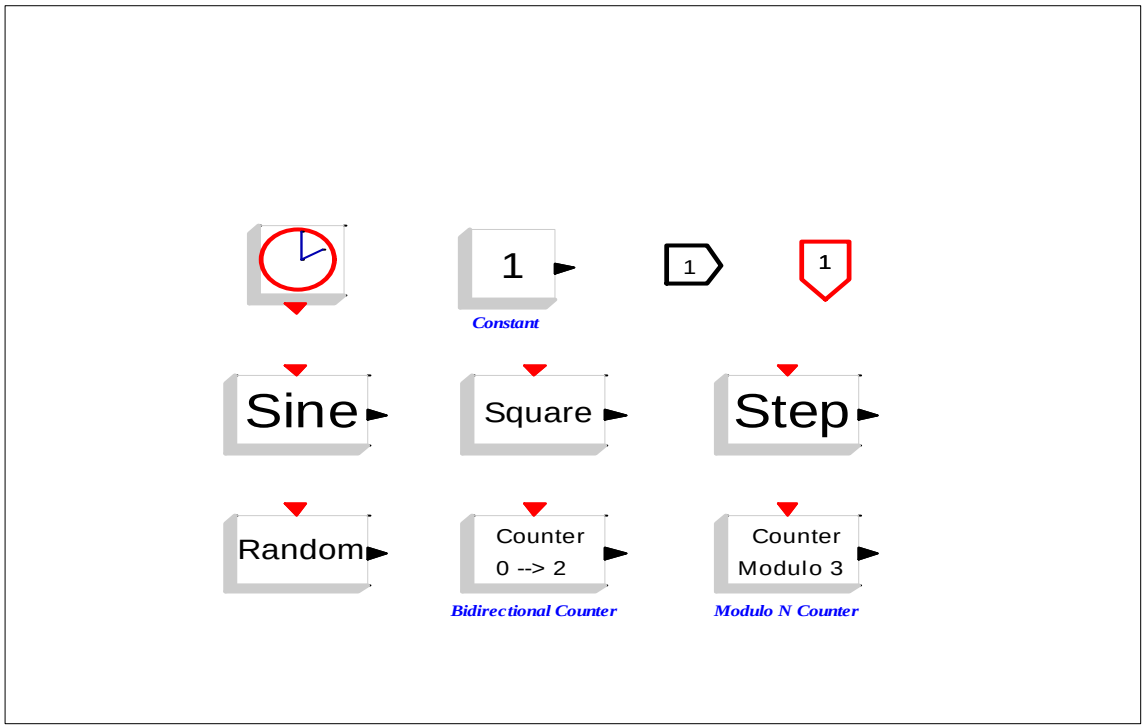
Linear-Palette



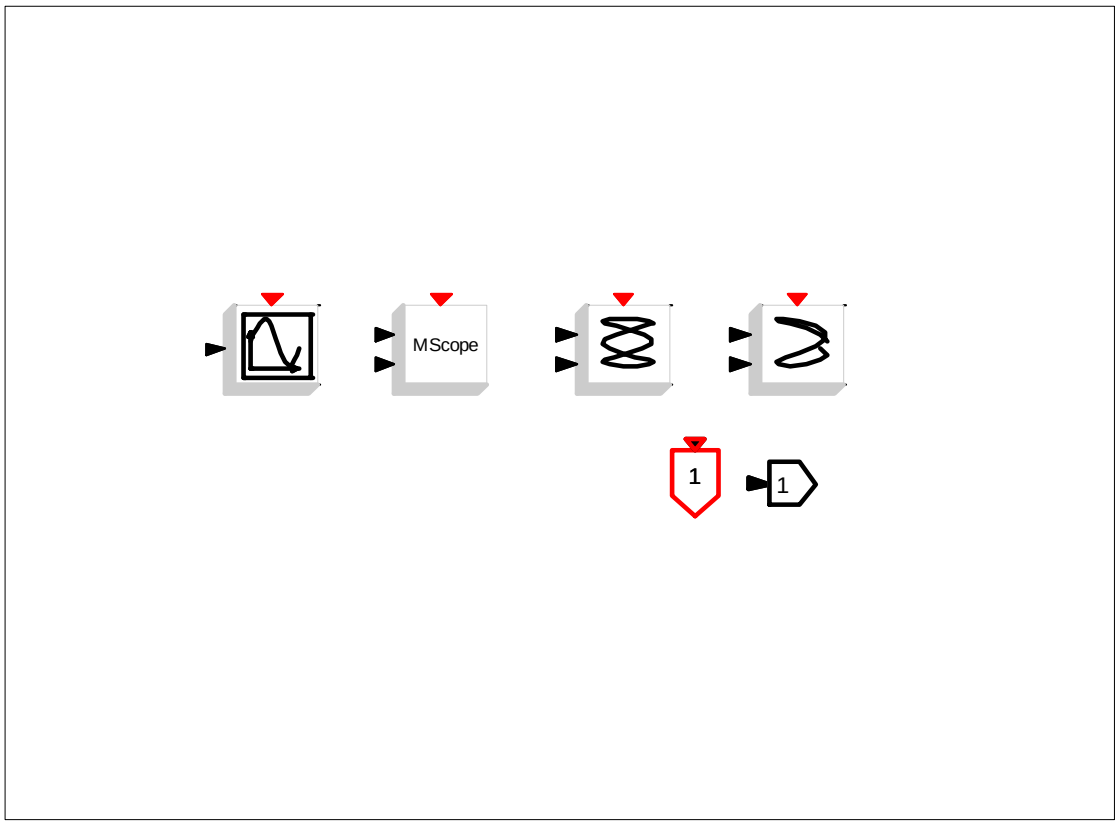
Die Non_Linear-Palette



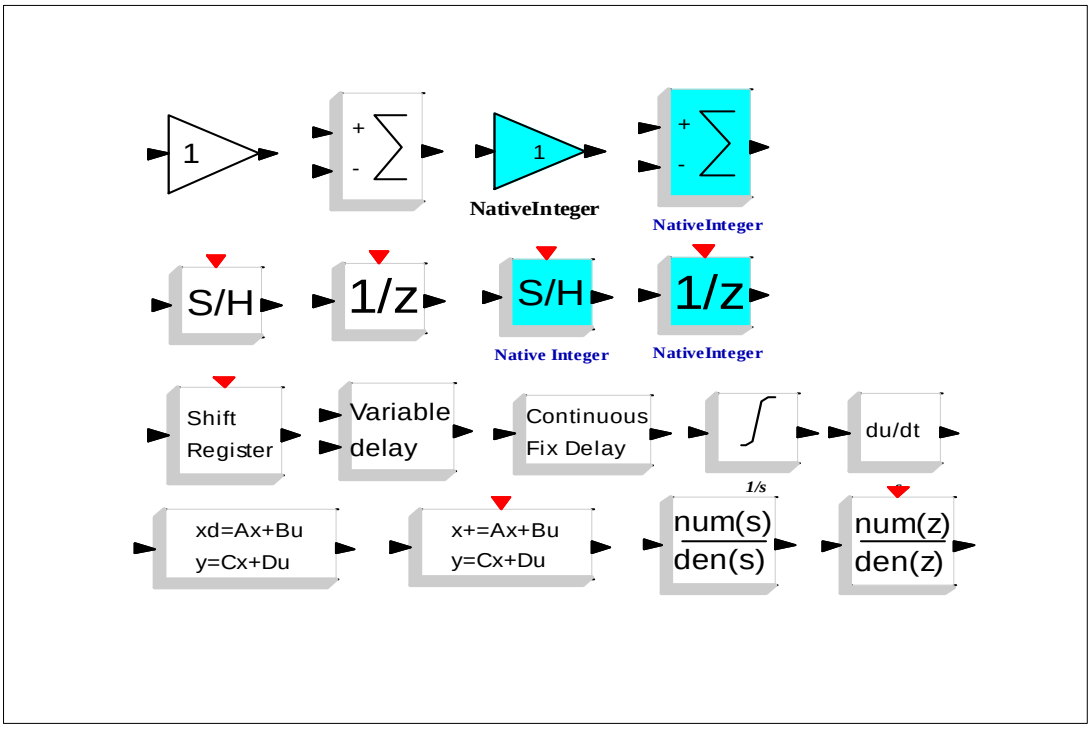
MCHP16_Source-Palette



MCHP16_Sinks- Palette



MCHP16_Linear- Palette



MCHP16_NonLinear- Palette