

Design Flow in DSP Projects

Stefan Schmitt

DSPepecialists GmbH, Wattstr. 11 - 13, 13355 Berlin, Germany

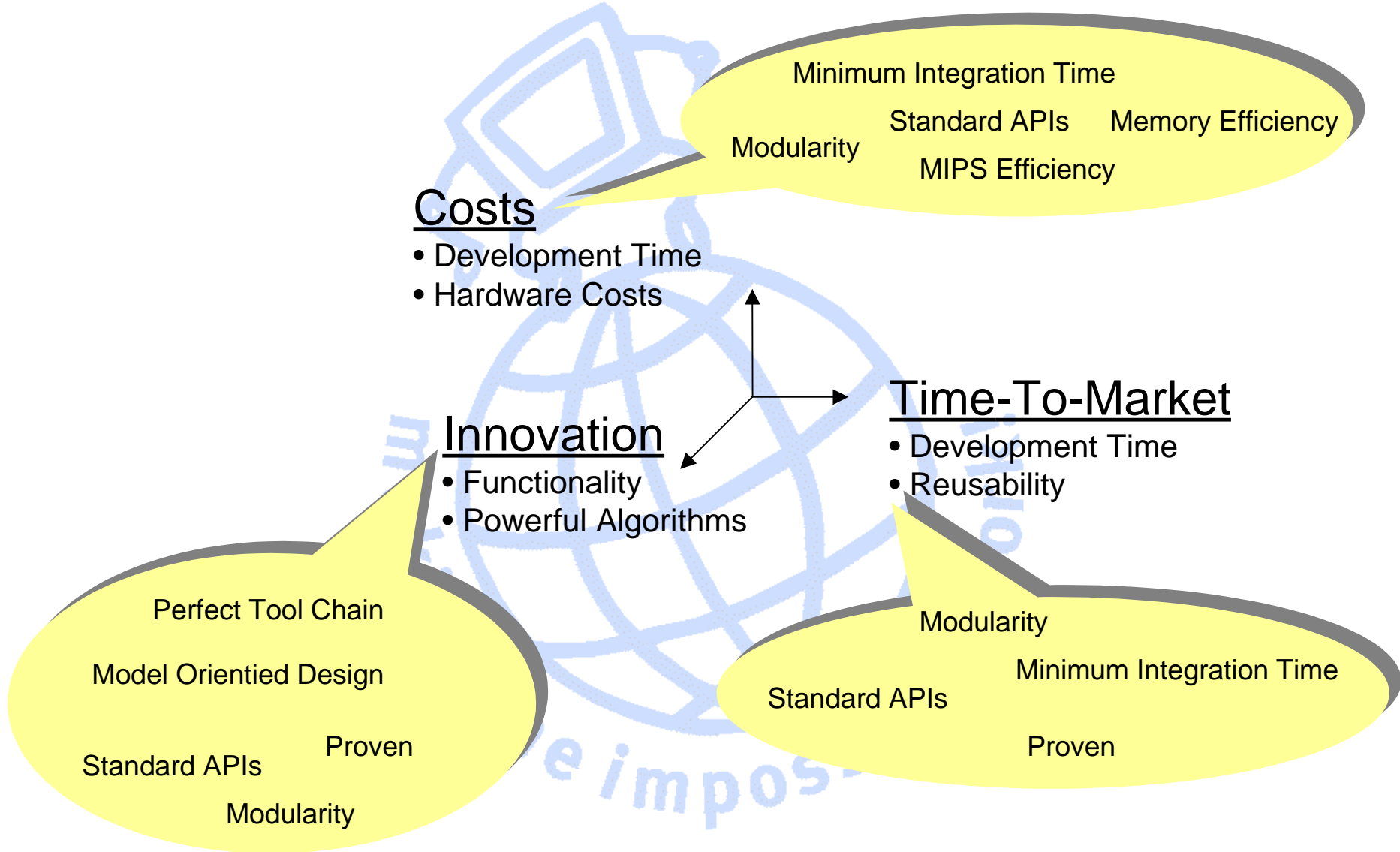
tel.: (+49) 30 / 467 805 - 0, fax.: (+49) 30 / 467 805 - 99

email: info@DSPepecialists.de, [www: www.DSPepecialists.de](http://www.DSPepecialists.de)

AGENDA

- DSP technology and requirements for optimal product developments
- Modular software design using assembly code
 - System software - application software
 - Object oriented assembly code design
- Design of efficient product code using of Simulink and RTW/TLC
 - Optimized design flows
 - Using Matlab/Simulink for generating target code
- Conclusion

Optimal Product Development Requirements



Advantages of Modular Software Design

- Separation of system software and application software
- OO-concept enables efficient and modular application
- Application software is hardware independent
- Integration of assembler and/or C code

Separation of System and Application

**Development Focus:
Application and Algorithms
(DSP-Libs)**

**Host Communication
Handler (DSP-HC)**

IO Handler (DSP-IO)

**System Start and
Initialization (DSPios)**

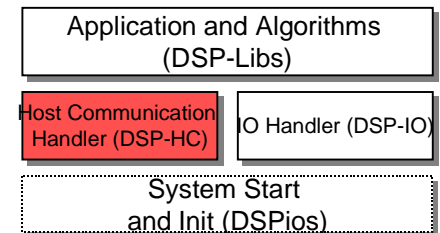
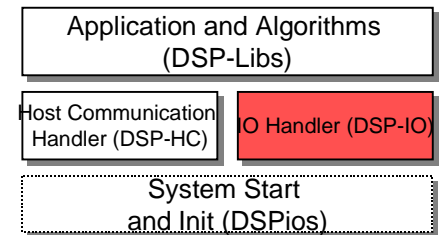
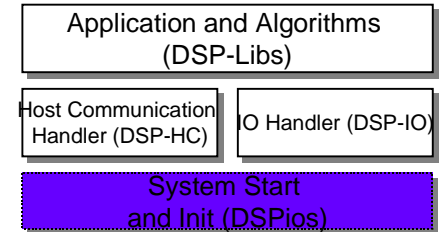


- Application is initialized and then runs in endless loop
- Application is loaded by ROM module or DSP-HC
- DSP-HC is started, thus Host can take over control of DSP system
- DSP-IO und DSP-HC are loaded by ROM module or DSP bootstrap
- Start and Initialization Code for System comes via DSP bootstrap (ROM, Host, IO, ...)

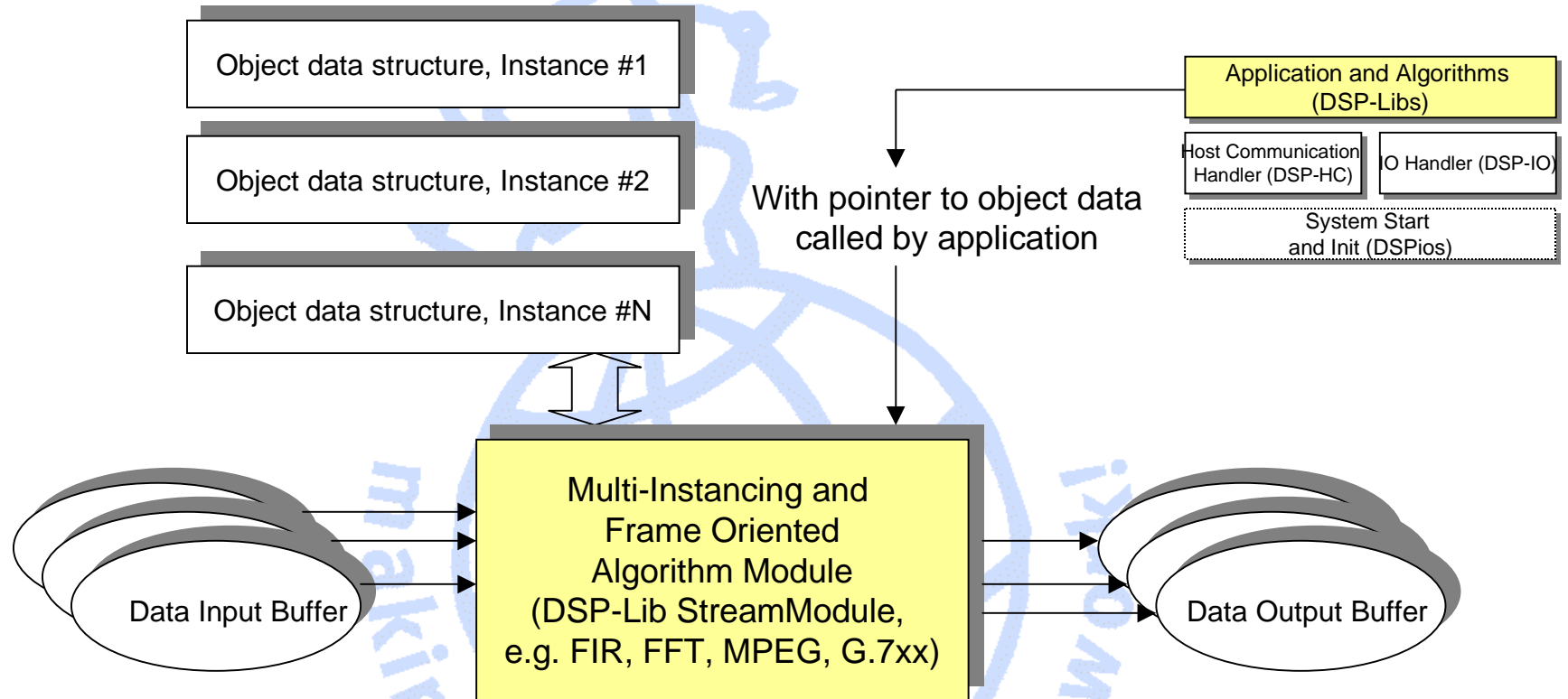
Program Space is deallocated after processed (RAM Saver Technology)

Advantages of System Software Separation

- System Startup:
 - System initialization and test (user extendable)
 - Stand alone operation (from any application source)
 - No RAM usage after processed
 - ⇒ Safe system startup with RAM Saver Technology
- Input Output Control
 - Seamless automatic data transfer
 - Automatic channel separation
 - Uniform API independent of peripheral hardware
 - Non-intrusive, DMA driven
 - ⇒ Application is independent of IO hardware
- Host Communication
 - Host controlled memory read/write and program start/stop
 - Bi-directional messaging
 - Uniform API independent of communication channel
 - Non-intrusive, DMA driven (PCI: < 0.1% performance loss)
 - ⇒ Application is independent of interfaces

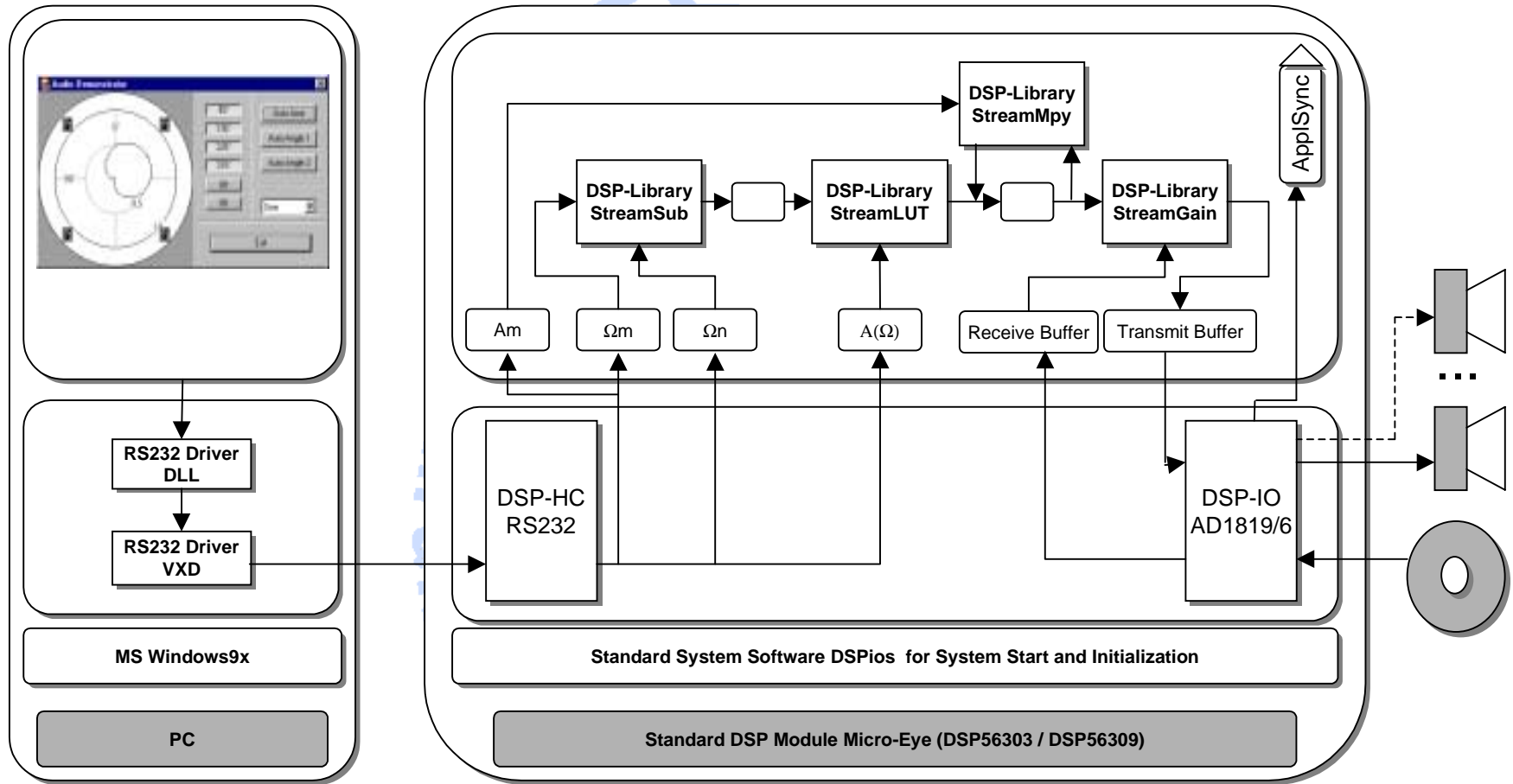


Object Oriented Design for Data Stream Processing



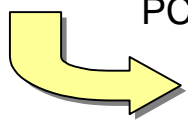
- Standardized calling interface (C compatible)
- Framewise processing of data streams for high runtime efficiency
- Object oriented: Multiple calls for Multi-Channel Processing
- Optimized assembler module for minimum code size and maximum efficiency
- Independent of object code compatible DSPs
- Software structure similar to models in graphical simulation tools (e.g. Simulink)

Software Structure „Dynamic Poly Gain Control“



PC System

DSP System



Audio Profile Based Dynamic Gain Control of n Channels

Application Main Program in Assembler or C

- Main Program (ASM)

```
MainProg
; --- initialize system
    Init

; --- start audio data transfer
    IO_Start    #IO_Struct

; --- enter mainloop
    DOR    FOREVER,MainLoop

        ApplSync
        FIR    #FIR_Left
        FIR    #FIR_Right

MainLoop

    EndProg
```

- Initialization Routine (ASM)

```
Init

; --- initialize FIR filter
SetStructBegin FIR_Left
SetStruct    FIR_Left,FIR_FrameCount,#>ApplFrameSize
            /FIR_FrameSize

SetStruct    FIR_Left,FIR_InBase,#AudioRecBuf_Base
SetStruct    FIR_Left,FIR_InSize,#AudioRecBuf_Size
SetStruct    FIR_Left,FIR_OutBase,#AudioTransBuf_Base
SetStruct    FIR_Left,FIR_OutSize,#AudioTransBuf_Size
SetStruct    FIR_Left,FIR_CoeffBase,#FIR_CoeffBuf_Base
SetStruct    FIR_Left,FIR_CoeffSize,#FIR_CoeffBuf_Size
SetStructEnd    FIR_Left

SetStructBegin FIR_Right
SetStruct    FIR_Right,FIR_FrameCount,#>ApplFrameSize
            /FIR_FrameSize

[...]

SetStruct    FIR_Right,FIR_CoeffSize,#FIR_CoeffBuf_Size
SetStructEnd    FIR_Right
```

- Main Program (C)

```
void main(void)
{
    Init();
    IO_Start();

    while(1)
    {
        ApplSync();
        FIR(&FIR_Left);
        FIR(&FIR_Right);
    }
}
```

- Initialization Routine (C)

```
Init()
{
    struct FIR{
        int FrameCount;
        int InBase;
        [...]
        int CoeffSize;
    }

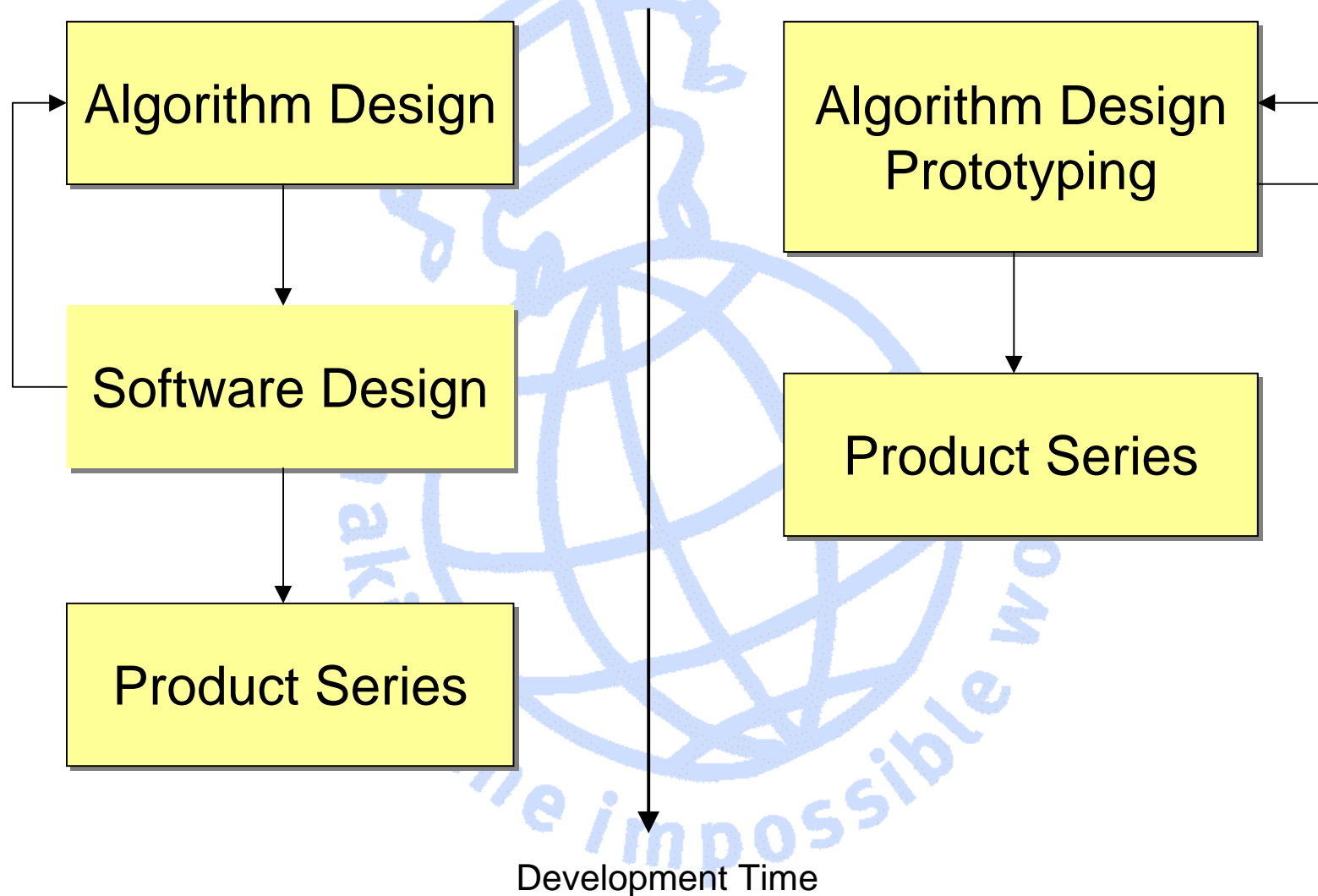
    FIR_Struct FIR_Left =
    {ApplFrameSize/FIR_FrameSize,
    AudioRecBuf_Base,[...],
    FIR_CoeffBuf_Size}

    [...]
}
```

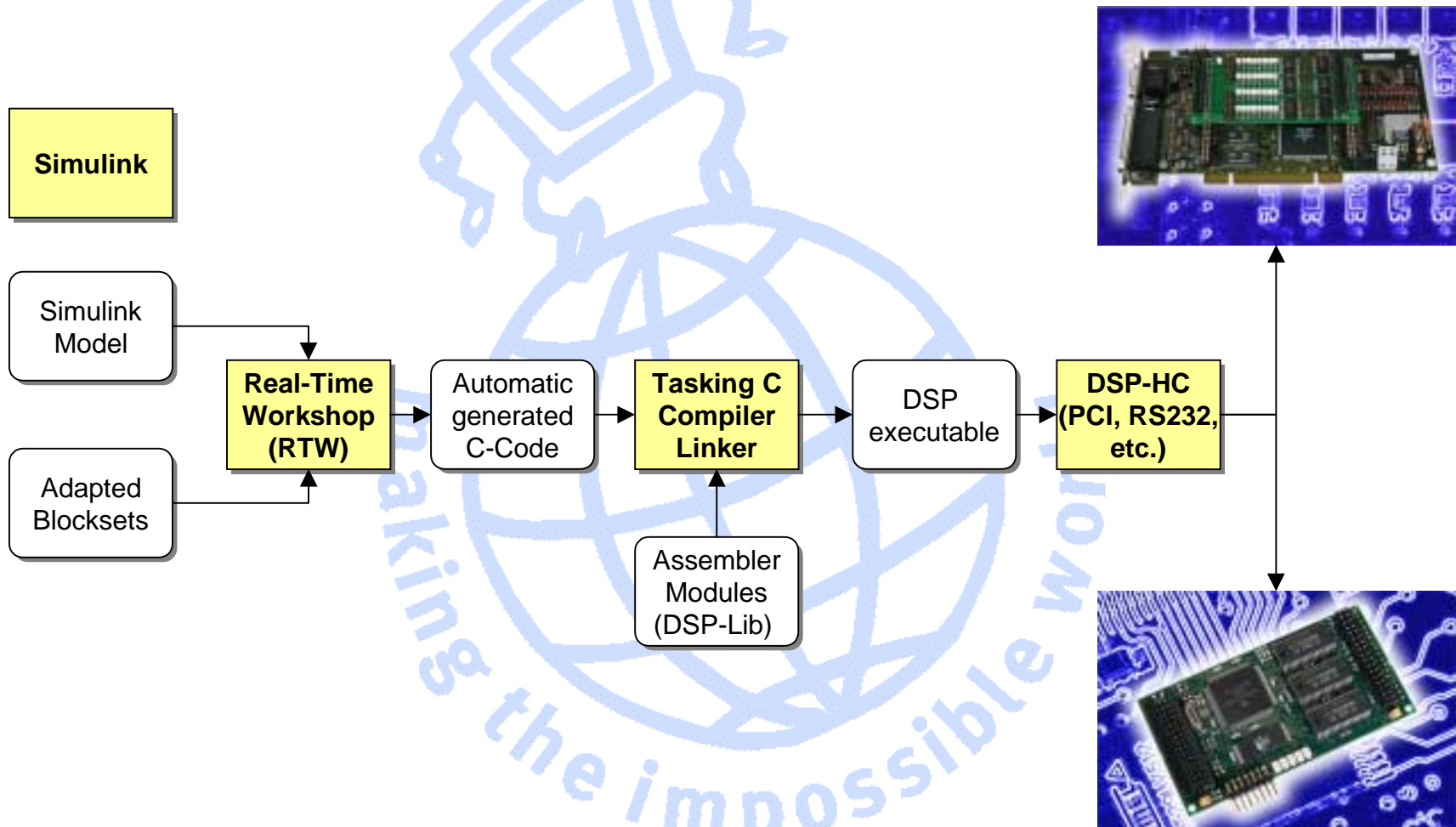
Optimizing Design Flow Using Graphical Tools

- Competing design flows
- Development process using Matlab/Simulink
- Generating executable target code using Simulink with Real Time Workshop (The Mathworks), Product Hardware Interface (DSPepecialists) and C Compiler (Tasking)
- Automatic generation of C code (Target Language Compiler process)
- Downloading and running DSP executable (Host Communication)

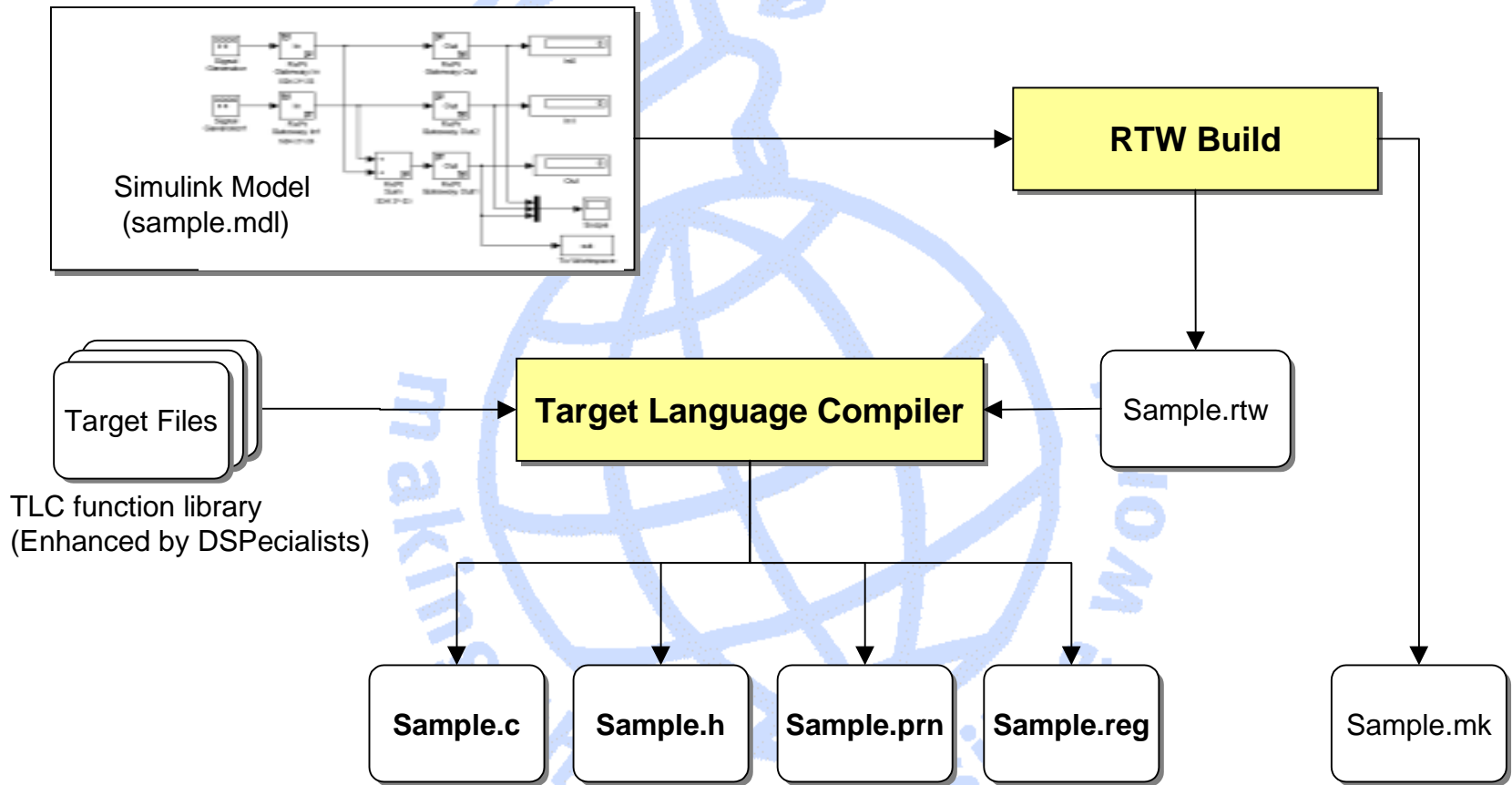
Competing DSP Software Design Flows



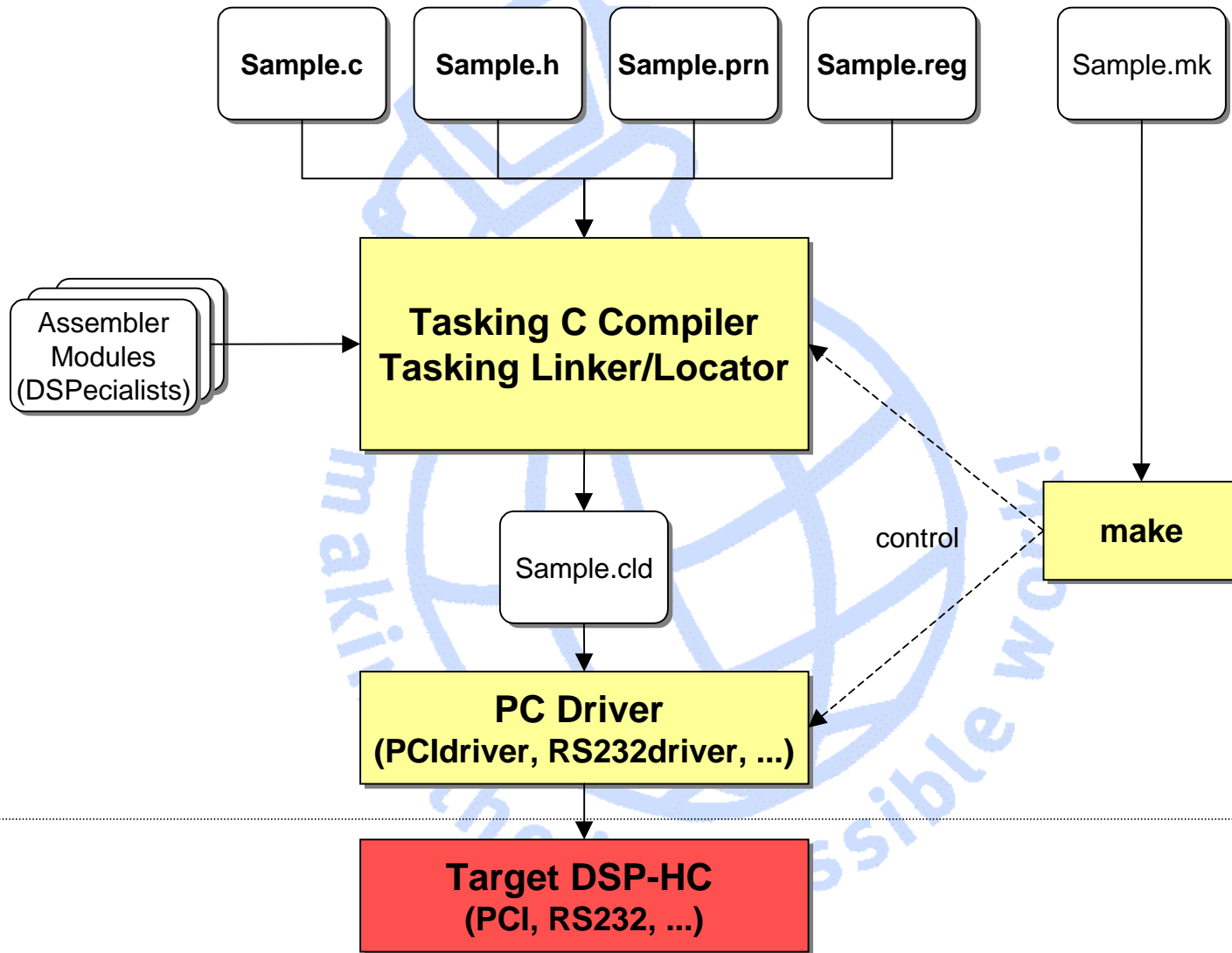
Tool Chain for Code Generation and Integration



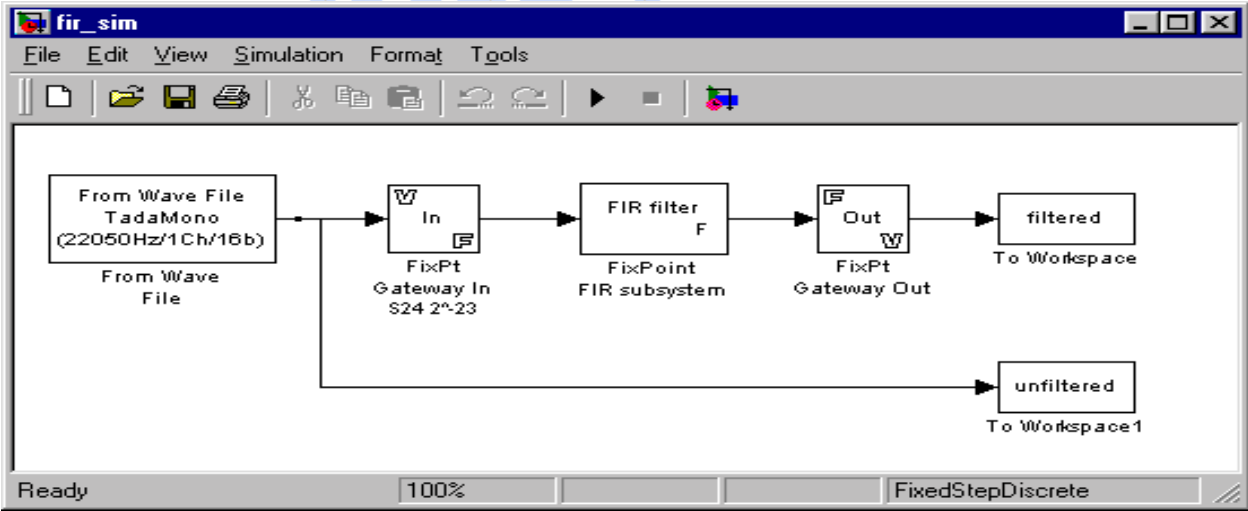
Automatic Generation of C Code with RTW



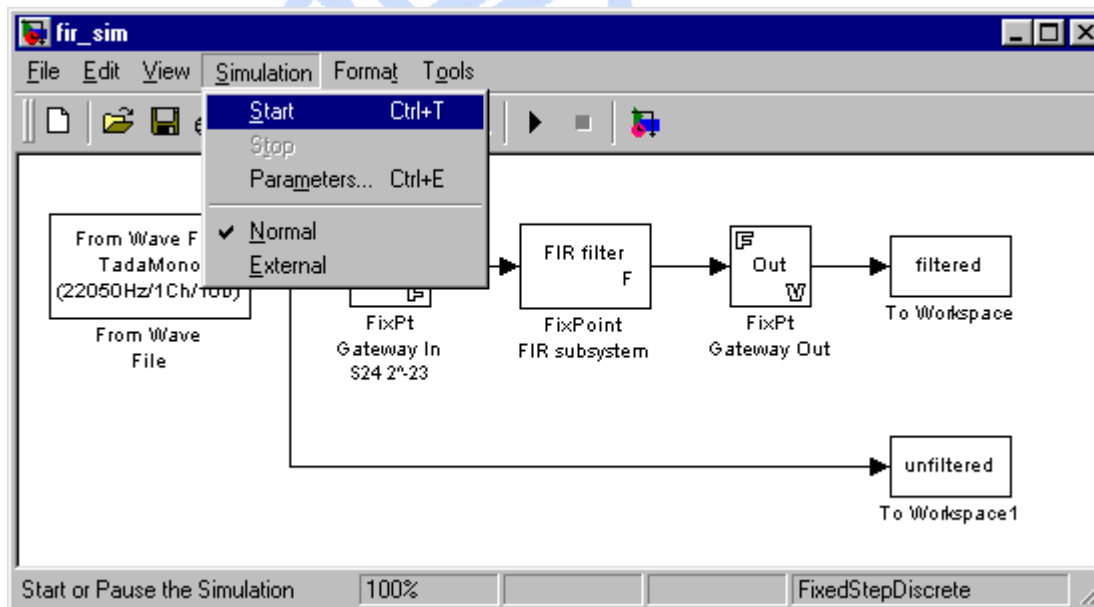
Automatic Target Integration



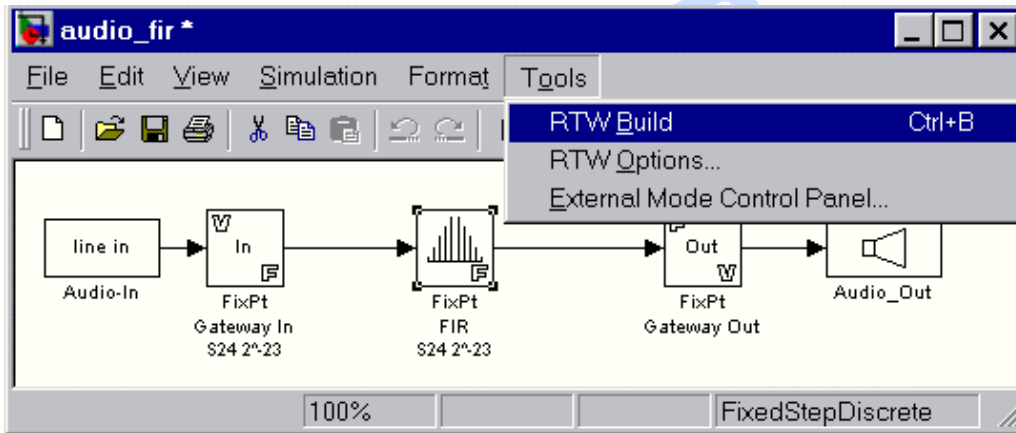
Example Project: FIR Filter for Audio Application



Example Project: Simulation



Example Project: Build and Download to Target



```

MATLAB Command Window
File Edit View Window Help
C:\PROGRAMME\Tasking\bin\as563.exe -OJNMS -RDRS -g audio_fir_new.src
DSP563xx/6xx assembler v2.1 r1 SN00085571-070 (c) 1998 TASKING, Inc.
### Linking ...
C:\PROGRAMME\Tasking\bin\cc563.exe -c1 -Wlk-w8 -Tpceye1_rs232 -M24 -L-LC:\PROGRAMME\Task
DSP563xx/6xx control program v2.1 r1 SN00085571-033 (c) 1998 TASKING, Inc.
### Created object audio_fir_new.out
### Locating ...
C:\PROGRAMME\Tasking\bin\lc563.exe -f4 -M -w8 -dpceye1_rs232.dsc -o audio_fir_new.cld au
DSP563xx/6xx locator v2.1 r1 SN00085571-031 (c) 1998 TASKING, Inc.
### Created executable audio_fir_new.cld
### Downloading audio_fir_new: gmake -f audio_fir_new.mk download
### Downloading audio_fir_new ...
perl E:\MATLAB\rtw\c\dspe\tools\down_drv.pl audio_fir_new.cld E:\MATLAB\rtw\c\dspe\tools

Do you want to download (d) or install (i) created model? (a) to abort download
Ready NUM

```

Conclusion

- Correct tool configuration enables optimal design flow
- Modular concept enables low development costs, fast time to market and high innovation rate
- Algorithm modeling/simulation and product code possible in one step
- Separation of system and application software enables focussing on central tasks
- DSPecialists services: Complete solutions for optimal DSP designs, Know-How in the fields of audio/video/telecommunications, optimal design methodologies