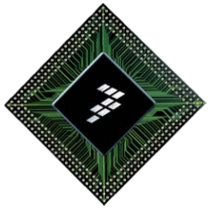
Freescale
Technology Forum

Collaboration. Innovation. Inspiration.

July, 2009



Getting Started With DSCs

John L. Winters
Senior Application Engineer

freescale™
semiconductor

Agenda

- FreeMASTER Overview
- Quick_Start Overview
- Processor Exert Overview
- Processor Expert Demo

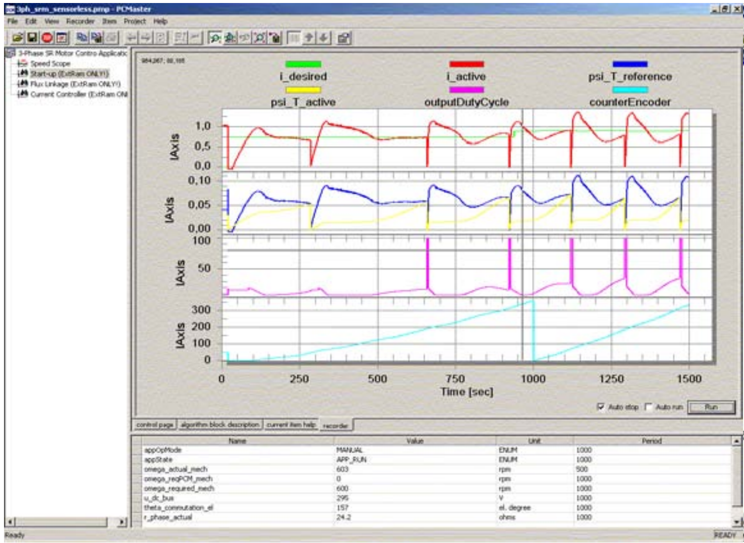
Agenda

- FreeMASTER Overview
- Quick_Start Overview
- Processor Exert Overview
- Processor Expert Demo

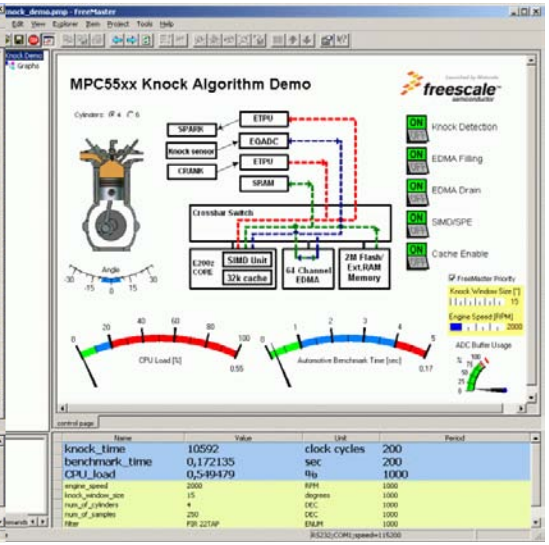
What is FREEMASTER?

- ▶ **Real-time Monitor**
- ▶ **Graphical Control Panel**
- ▶ **Demonstration Platform & Selling Tool**

→ **FOR YOUR EMBEDDED APPLICATION**

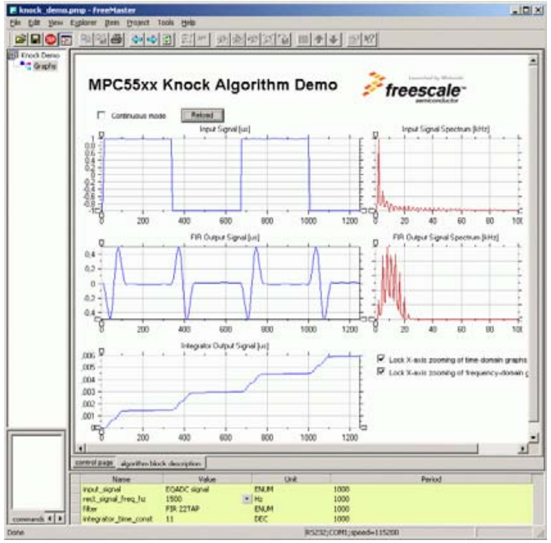


Name	Value	Unit	Period
appOpMode	MANUAL	ENUM	1000
appState	APP_RUN	ENUM	1000
omega_actual_mech	603	rpm	500
omega_reqPCM_mech	0	rpm	1000
omega_required_mech	600	rpm	1000
u_dc_bus	295	V	1000
theta_commutation_el	157	el. degree	1000
r_phase_actual	24.2	ohms	1000



Name	Value	Unit	Period
knock_time	10592	clock cycles	200
benchmark_time	0,172135	sec	200
CPU_load	0,549479	%	1000
engine_speed	2000	RPM	1000
knock_window_size	15	degrees	1000
num_of_cylinders	4	DEC	1000
num_of_samples	250	DEC	1000
filter	FIR 22TAP	ENUM	1000

RS232;COM1;speed=115200



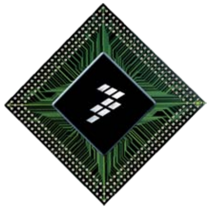
knock_demo.pmp - FreeMaster
File Edit View Explorer Item Project Tools Help
Knock Demo
Graphs
MPC55xx Knock Algorithm Demo
freescale semiconductor
□ Continuous mode
Reload
Input Signal [us]
Input Signal Spectrum [kHz]
FIR Output Signal [us]
FIR Output Signal Spectrum [kHz]
Integrator Output Signal [us]
☑ Lock X-axis zooming of time domain graphs
☑ Lock X-axis zooming of frequency-domain graphs
control page
algorithm block description
Done

Name	Value	Unit	Period
input_signal	EQADC signal	ENUM	1000
rect_signal_freq_hz	1500	Hz	1000
filter	FIR 22TAP	ENUM	1000
integrator_time_const	11	DEC	1000

RS232;COM1;speed=115200

FREEMASTER

As a Real-time Monitor



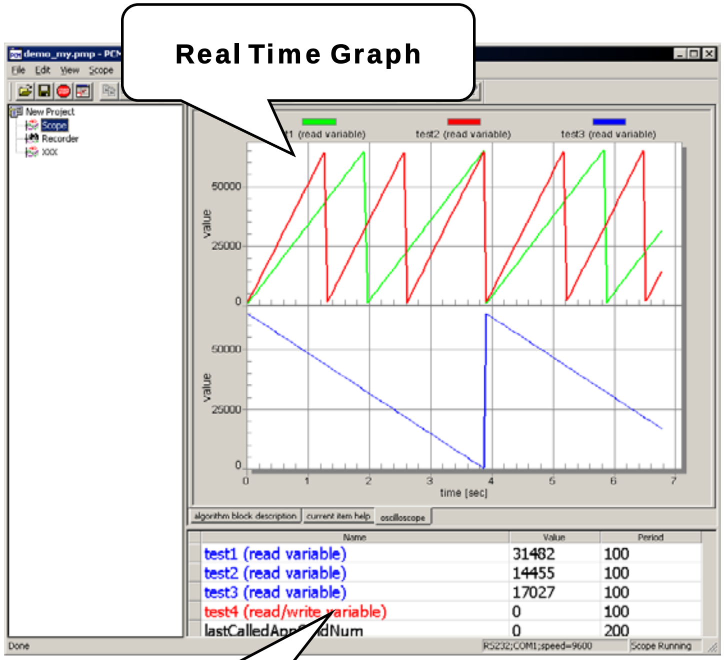
FREEMASTER as a Real-time Monitor

- Connects to an embedded application
  - Natively by SCI, UART
  - JTAG/EOnCE (56F8xxx only)
  - BDM (HCS08, HCS12 only)
  - CAN Calibration Protocol, custom CAN protocol
  - Ethernet, TCP/IP
  - Any of the above remotely over the network

- Enables access to application memory
  - Parses ELF application executable file
  - Parses DWARF debugging information in the ELF file
  - Knows addresses of global and static C-variables
  - Knows variable sizes, structure types, array dimensions, etc…

FREEMASTER as a Real-time Monitor

- ▶ Displays the variable values in a range of formats
  - Text, tabular grid
    - Variable name
    - Value as hex, dec or bin number
    - Min. / max. values
    - Number-to-text labels
  - Real-time waveforms
    - Up to 8 variables simultaneously in an oscilloscope-like graph
  - High-speed recorded data
    - Up to 8 variables in on-board memory transient recorder



Real Time Graph

Name	Value	Period
test1 (read variable)	31482	100
test2 (read variable)	14455	100
test3 (read variable)	17027	100
test4 (read/write variable)	0	100
lastCalledApp[illegible]dNum	0	200

Variable Watch

FREEMASTER as a Real-time Monitor

- Additional features
  - Variable transformations
    - Variable value can be transformed to the custom unit
    - Variable transformations may reference other variable values
    - Values are transformed back when writing a new value to the variable
  - Application commands
    - Command code and parameters are delivered to an application for arbitrary processing
    - After processed (asynchronously to a command delivery) the command result code is returned to the PC
  - Ability to protect memory regions
    - Describing variables visible to FreeMASTER
    - Declaring variables as read-write to read-only for FreeMASTER
      - Access is guarded by the embedded-side driver

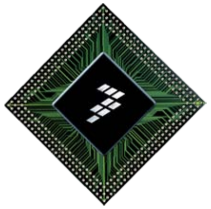
FREEMASTER as a Real-time Monitor

▶ Highlights

- FreeMASTER helps developers to debug or tune their applications
- Replaces debugger in situations when the processor core can not be simply stopped (i.e. motor control)
- Recorder may be used to visualize transitions in near 10-us resolution

FREEMASTER

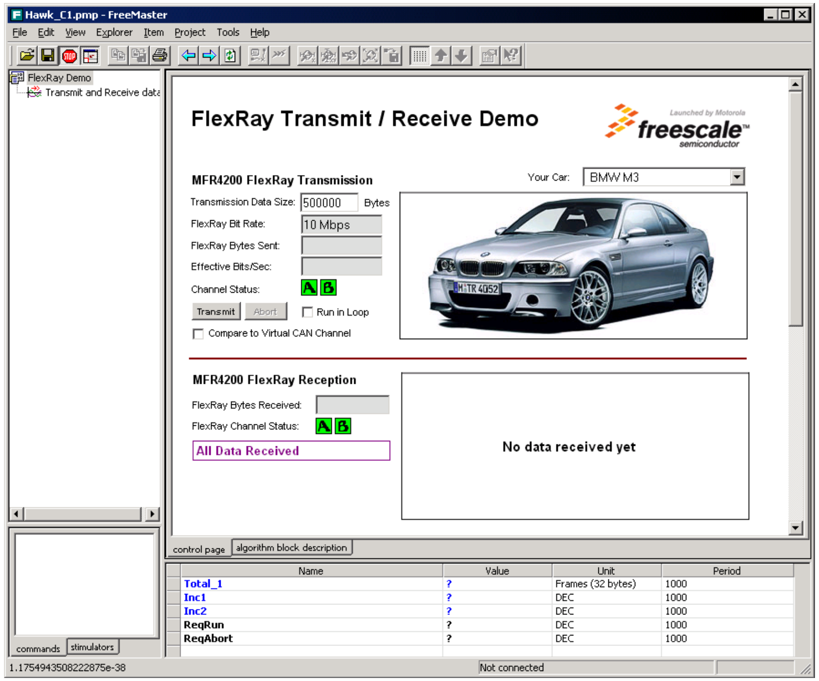
As a GUI for your Embedded Application



FREEMASTER as a Graphical User Interface

- ►Using FreeMASTER as a Graphical Control Panel
  - Variable Watch pane enables direct setting of the variable value
  - Sending Application Commands from the application GUI
  - Time-table stimulation of the variable value

  - HTML Pages and Forms
    - JScript or VBScript
    - Push buttons
    - Images, indicators
    - Sounds, videos
    - Sliders, gauges and other 3rd party ActiveX controls



Hawk_C1.pmp - FreeMaster
File Edit View Explorer Item Project Tools Help
FlexRay Demo
Transmit and Receive data
Transmit
Abort
control page
algorithm block description
commands
stimulators

FlexRay Transmit / Receive Demo

MFR4200 FlexRay Transmission

Your Car: BMW M3

Transmission Data Size: 500000 Bytes

FlexRay Bit Rate: 10 Mbps

FlexRay Bytes Sent:

Effective Bits/Sec:

Channel Status: A B

□ Run in Loop

□ Compare to Virtual CAN Channel

MFR4200 FlexRay Reception

FlexRay Bytes Received:

FlexRay Channel Status: A B

All Data Received

No data received yet

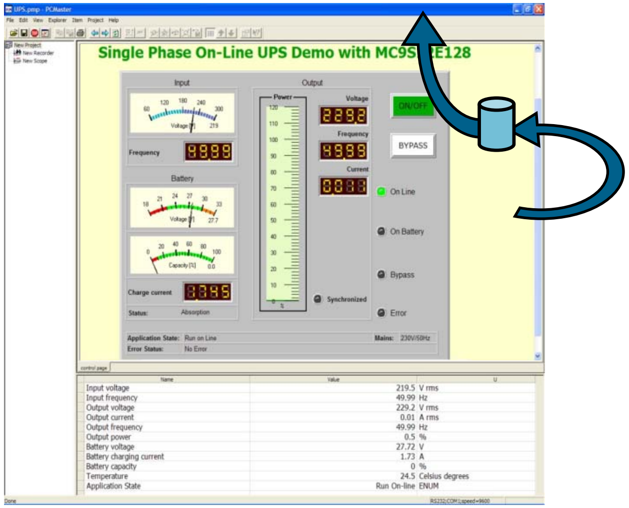
Name	Value	Unit	Period
Total_1	?	Frames (32 bytes)	1000
Inc1	?	DEC	1000
Inc2	?	DEC	1000
ReqRun	?	DEC	1000
ReqAbort	?	DEC	1000

1.1754943508222875e-38

Not connected

FREEMASTER as a Graphical User Interface

- ► Scripting in FreeMASTER
  - HTML pages are displayed directly in the FreeMASTER window
  - HTML may contain scripts and ActiveX objects
    - FreeMASTER itself implements an invisible ActiveX object
    - Script accesses the FreeMASTER functionality through this object
      - Variable access
      - Stimulator access
      - Application Commands
      - Recorder Data



Status: Absorption

Application State: Run on Line

Mains: 230V/50Hz

Error Status: No Error

Name	Value	U
Input voltage	219.5	V rms
Input frequency	49.99	Hz
Output voltage	229.2	V rms
Output current	0.01	A rms
Output frequency	49.99	Hz
Output power	0.5	%
Battery voltage	27.72	V
Battery charging current	1.73	A
Battery capacity	0	%
Temperature	24.5	Celsius degrees
Application State	Run On-line	ENUM

Done

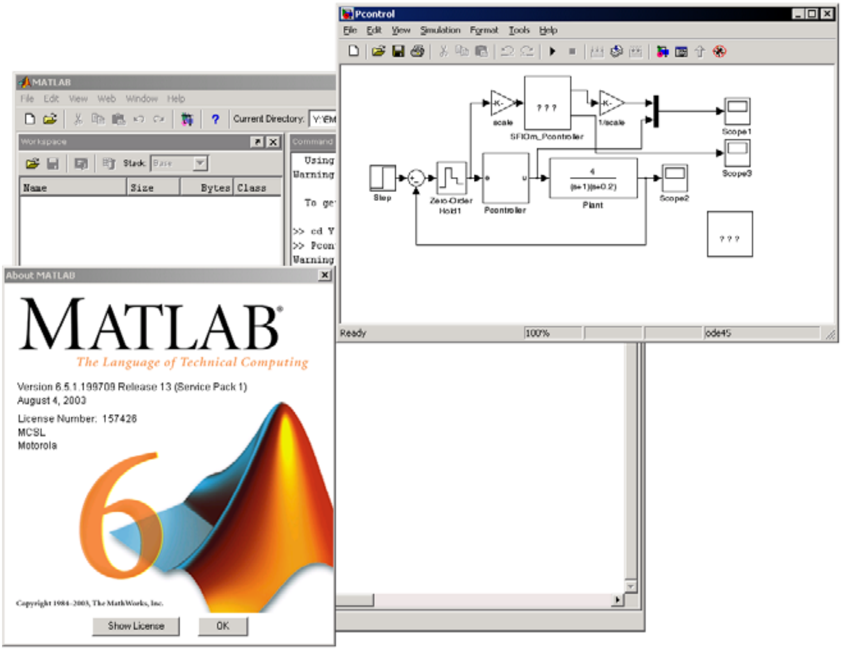
RS232;COM1;speed=9600

- HTML may host whole applications, for example Excel
  - Excel Visual Basic macros may access FreeMASTER as well

FREEMASTER as a Graphical User Interface

► Target-in-loop Simulations

- FreeMASTER invisible ActiveX object is accessible also by external standalone applications
  - Standard C++ or VB applications
  - Excel & Visual Basic for Applications
  - Matlab, Simulink
- Target-in-loop Simulation
  - Matlab or Simulink engine lets embedded application to perform calculations



About MATLAB

MATLAB®

The Language of Technical Computing

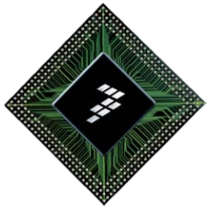
Version 6.5.1.199709 Release 13 (Service Pack 1)
August 4, 2003

License Number: 157428
MCSL
Motorola

Copyright 1984-2003, The MathWorks, Inc.

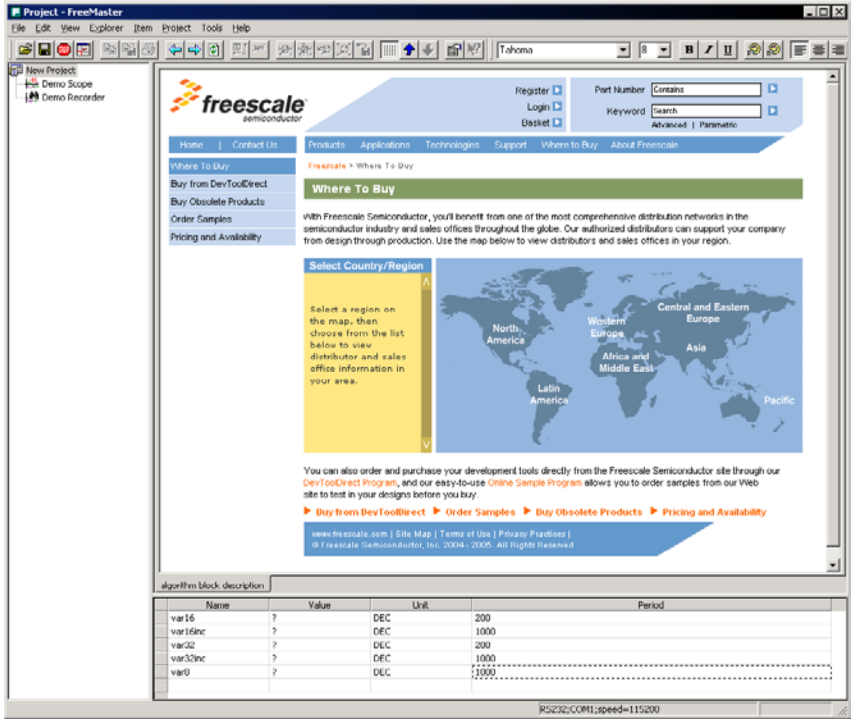
FREEMASTER

As a Selling Tool



FREEMASTER as a Selling Tool

- ► FreeMASTER helps Freescale marketers to sell our work
  - FreeMASTER project can visualize any detail of how the embedded application works
  - HTML Pages embed text images, videos together with live application data
  - FreeMASTER acts as a web-browser so it is possible to navigate to online shop directly without even leaving a FreeMASTER environment
  - FreeMASTER helps Freescale customers to sell their work



Where To Buy

With Freescale Semiconductor, you'll benefit from one of the most comprehensive distribution networks in the semiconductor industry and sales offices throughout the globe. Our authorized distributors can support your company from design through production. Use the map below to view distributors and sales offices in your region.

Select a region on the map, then choose from the list below to view distributor and sales office information in your area.

You can also order and purchase your development tools directly from the Freescale Semiconductor site through our DevToolDirect Program, and our easy-to-use Online Sample Program allows you to order samples from our Web site to test in your designs before you buy.

▶ Buy from DevToolDirect ▶ Order Samples ▶ Buy Obsolete Products ▶ Pricing and Availability

www.freescale.com | Site Map | Terms of Use | Privacy Practices |
© Freescale Semiconductor, Inc. 2004 - 2005. All Rights Reserved

Name	Value	Unit	Period
var16	?	DEC	200
var16inc	?	DEC	1000
var32	?	DEC	200
var32inc	?	DEC	1000
var8	?	DEC	1000

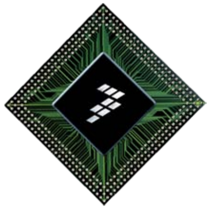
RS232;COM1;speed=115200

FREEMASTER as a Selling Tool

- FreeMASTER is Free!
  - The FreeMASTER is freely available from the Freescale web
  - License agreement prevents using FreeMASTER with processors from competition
  - Free redistribution enables Freescale customers to pack FreeMASTER with their products

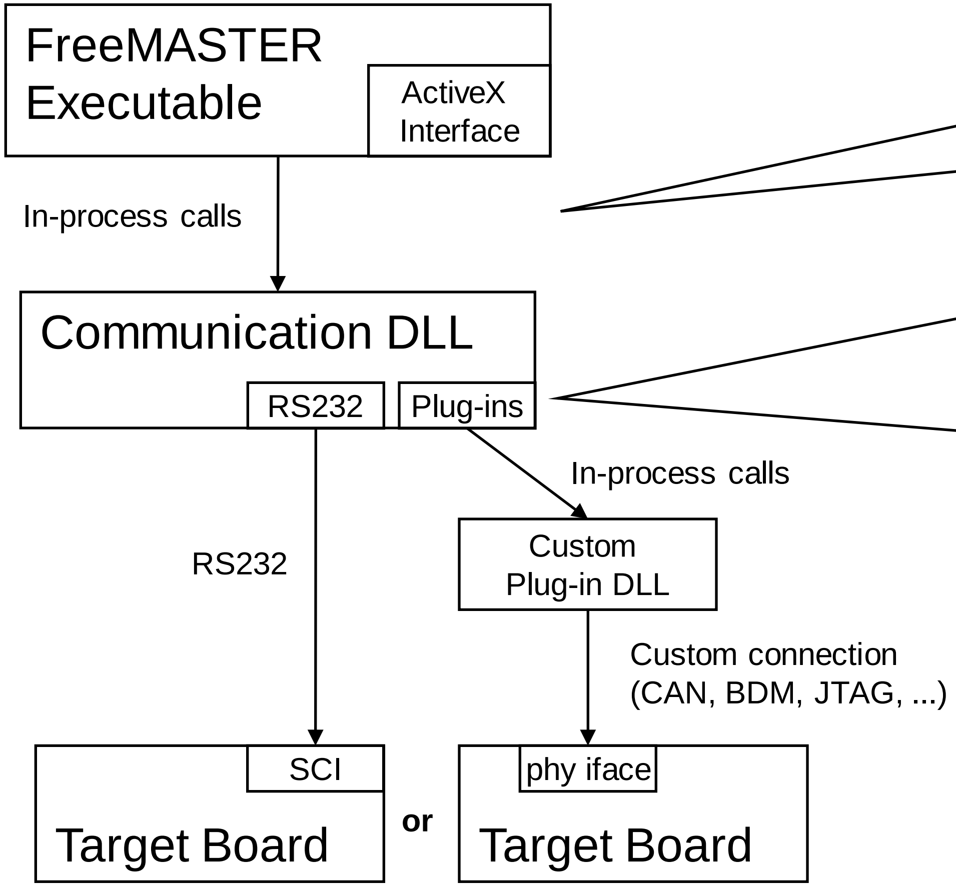
http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=FREEMASTER

Inside FreeMASTER Application



Internal Application Structure

▶ Basic FreeMASTER Communication Diagram



So-called in-process calls are the simple calls to functions located in dynamically loaded libraries (DLLs). The calls are fast just like if the function would be located inside the executable itself.

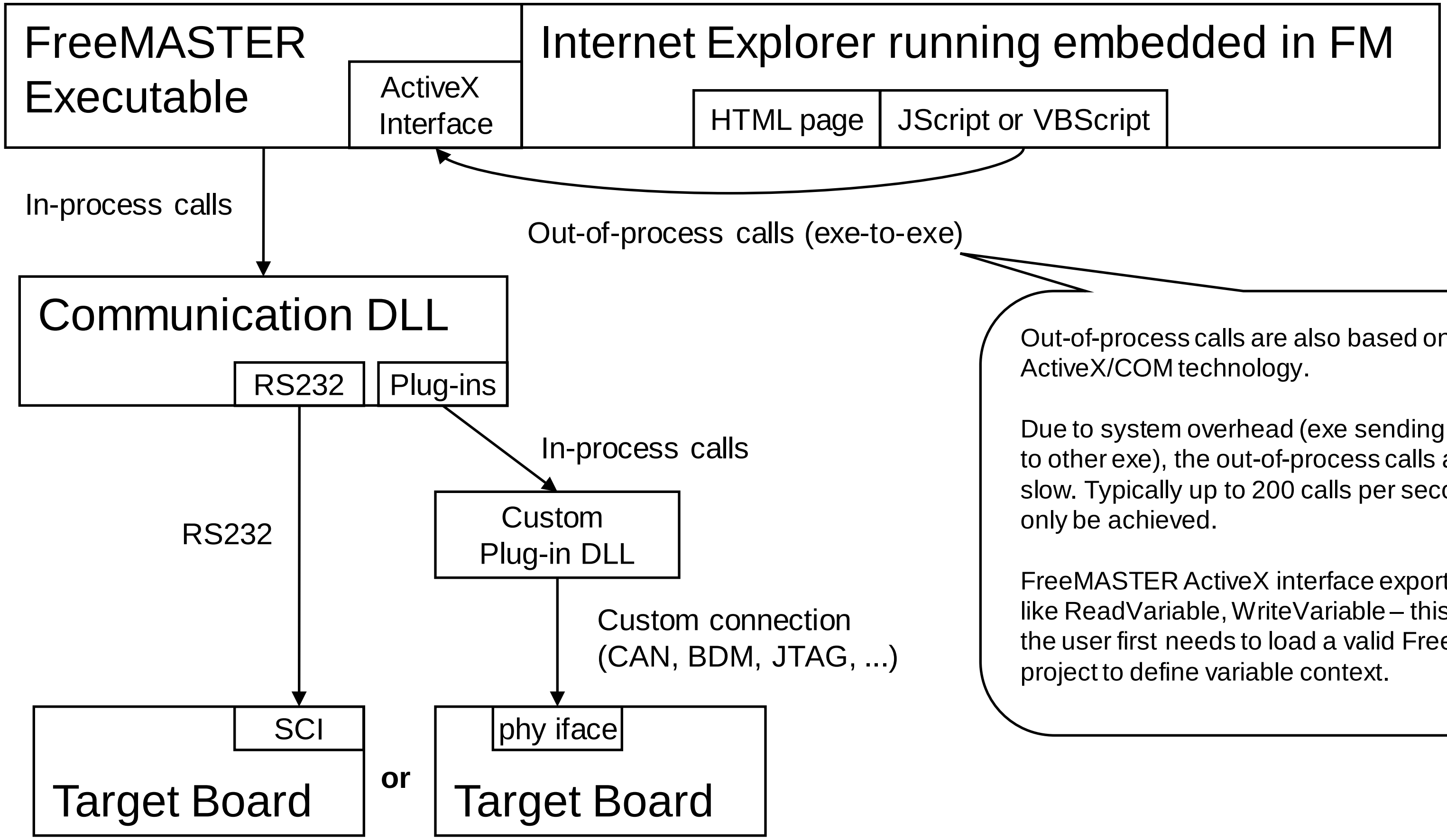
FreeMASTER enables to use custom plug-in modules to implement the communication layer.

We deliver few plug-ins in the standard FreeMASTER distribution. CAN, BDM and other connections are possible, not always fully featured. For example BDM only allows memory reads and writes, not the recorder or TSA feature.

FreeMASTER plug-ins use Microsoft COM (ActiveX) procedure call standard. Typically the fast in-process plug-in DLLs are made.

Internal Application Structure

- FreeMASTER Communication with HTML/JScript Pages



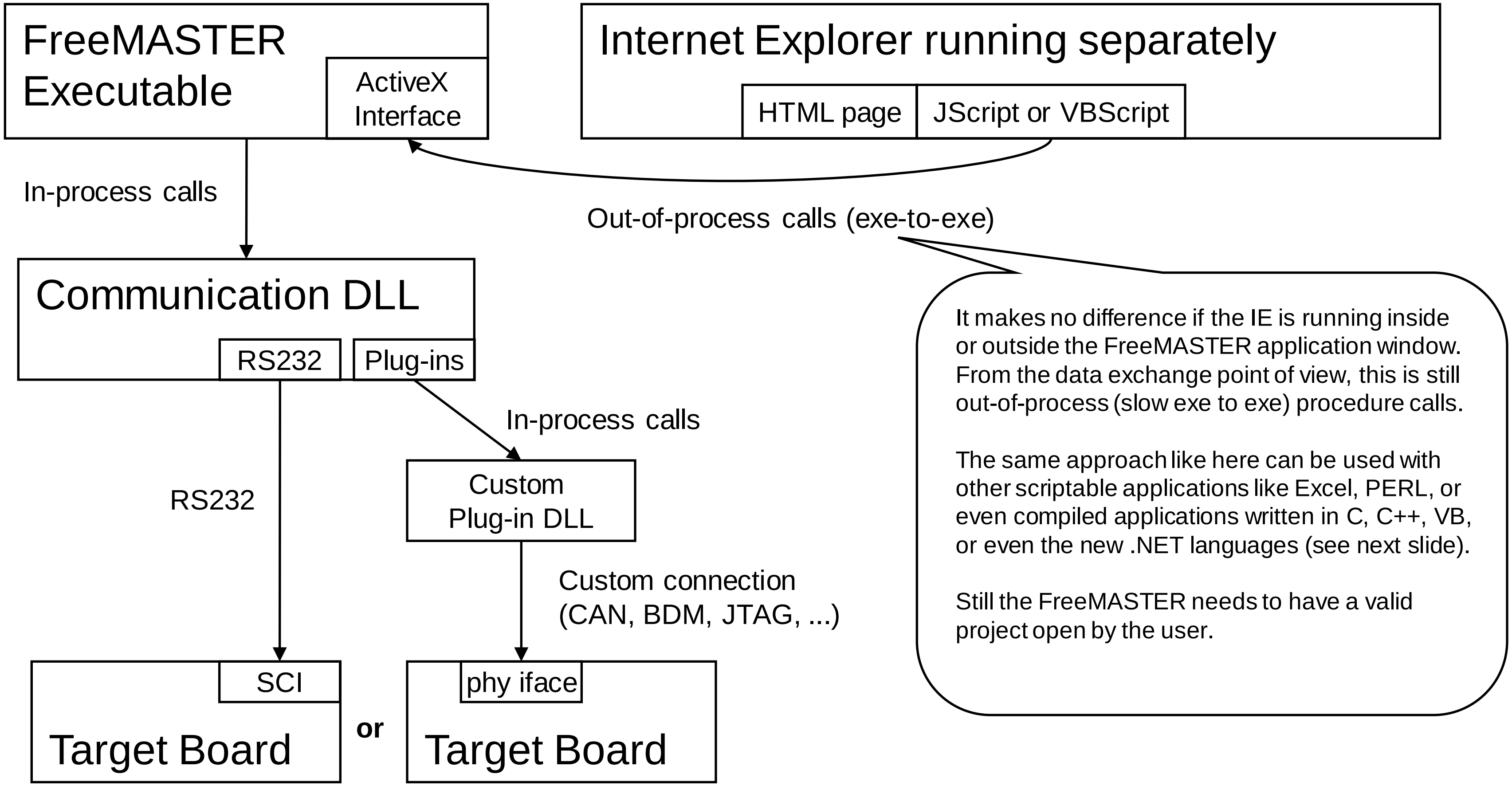
Out-of-process calls are also based on Microsoft ActiveX/COM technology.

Due to system overhead (exe sending messages to other exe), the out-of-process calls are quite slow. Typically up to 200 calls per seconds can only be achieved.

FreeMASTER ActiveX interface exports methods like ReadVariable, WriteVariable – this means the user first needs to load a valid FreeMASTER project to define variable context.

Internal Application Structure

- Internet Explorer Running Separately (no difference)



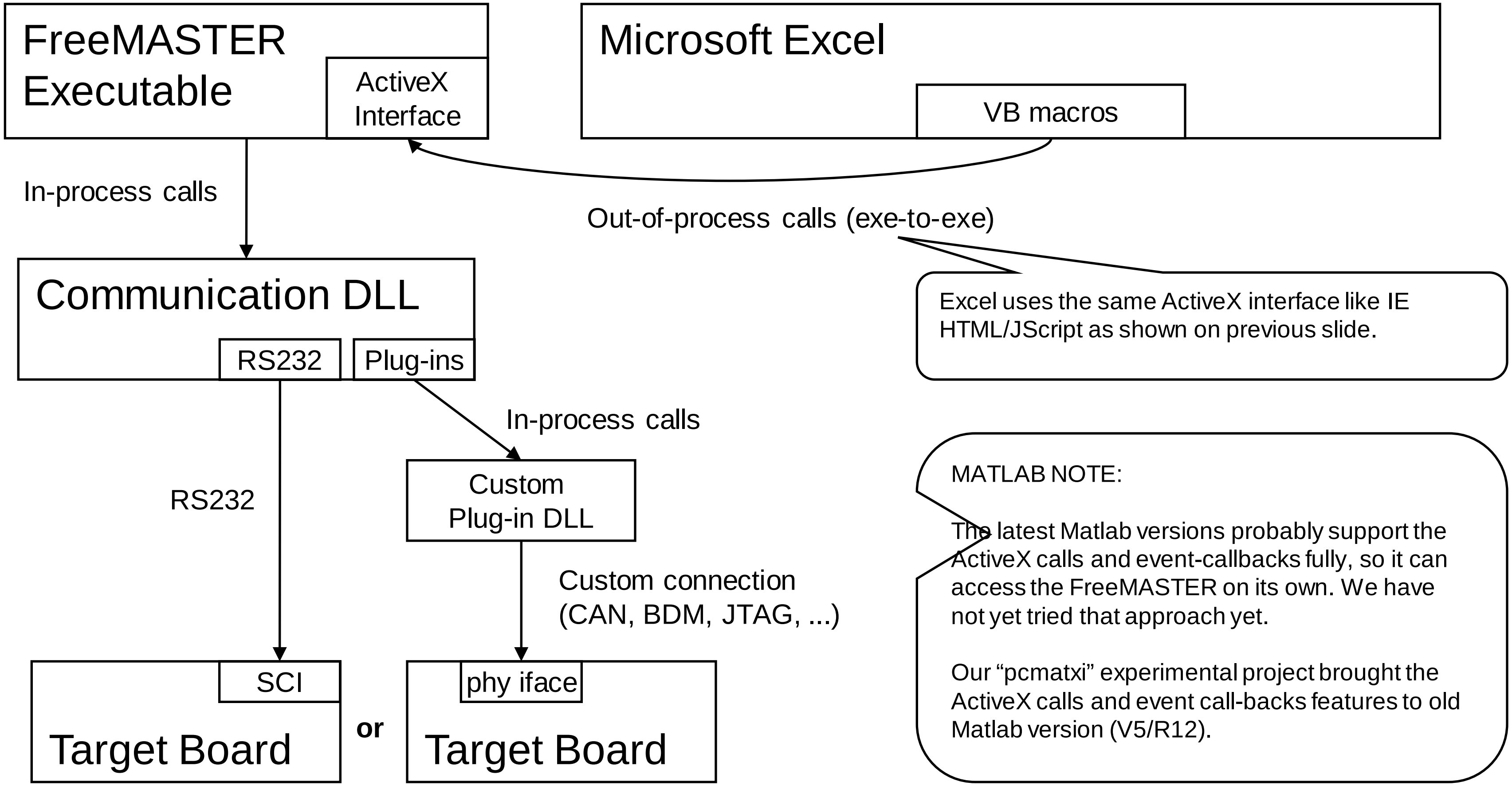
It makes no difference if the IE is running inside or outside the FreeMASTER application window. From the data exchange point of view, this is still out-of-process (slow exe to exe) procedure calls.

The same approach like here can be used with other scriptable applications like Excel, PERL, or even compiled applications written in C, C++, VB, or even the new .NET languages (see next slide).

Still the FreeMASTER needs to have a valid project open by the user.

Internal Application Structure

- ►Excel (or other application) accessing FM ActiveX



Excel uses the same ActiveX interface like IE HTML/JScript as shown on previous slide.

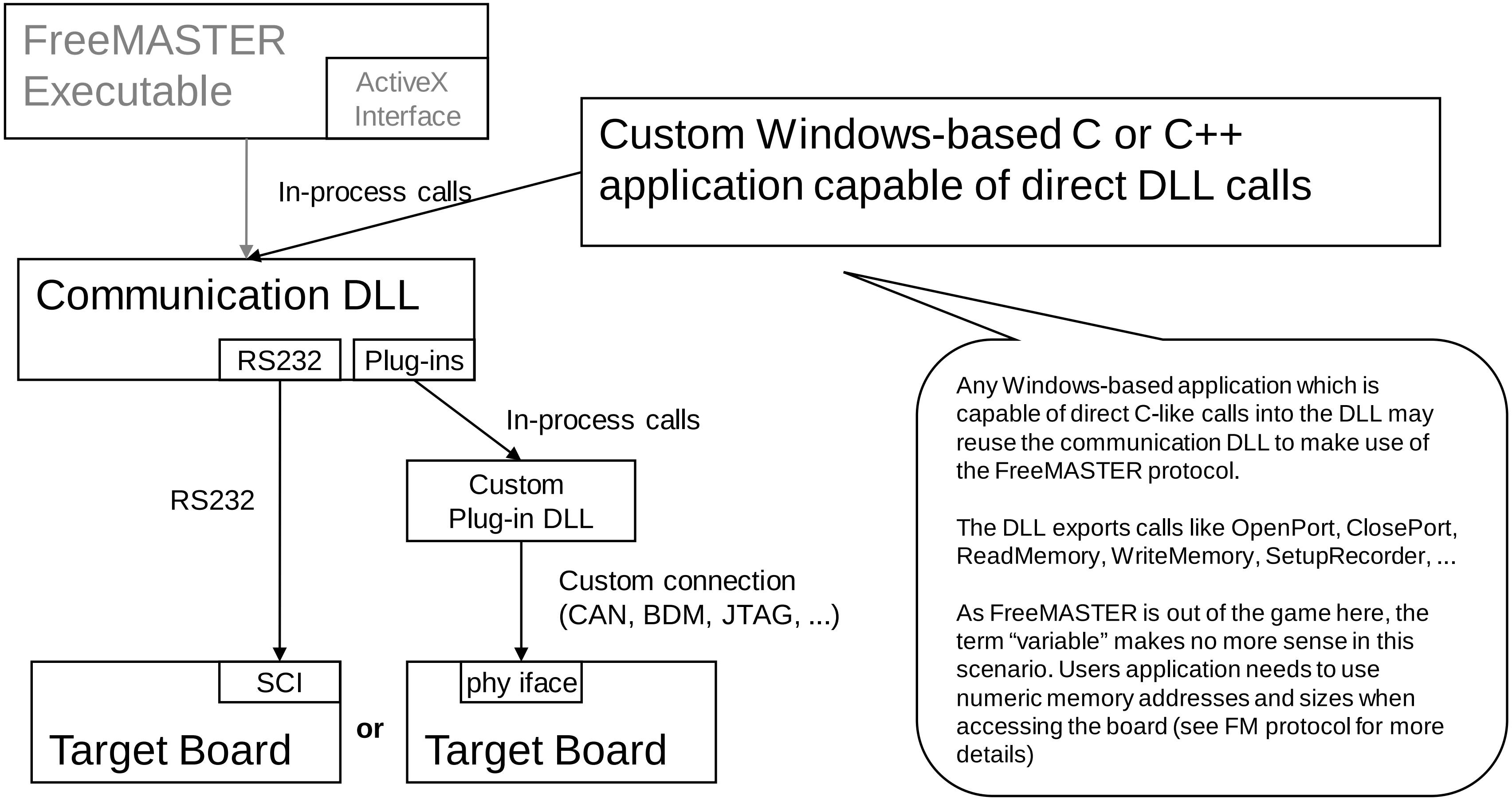
MATLAB NOTE:

The latest Matlab versions probably support the ActiveX calls and event-callbacks fully, so it can access the FreeMASTER on its own. We have not yet tried that approach yet.

Our “pcmatxi” experimental project brought the ActiveX calls and event call-backs features to old Matlab version (V5/R12).

Internal Application Structure

- Other Ways to Access Target Microprocessor: C, C++



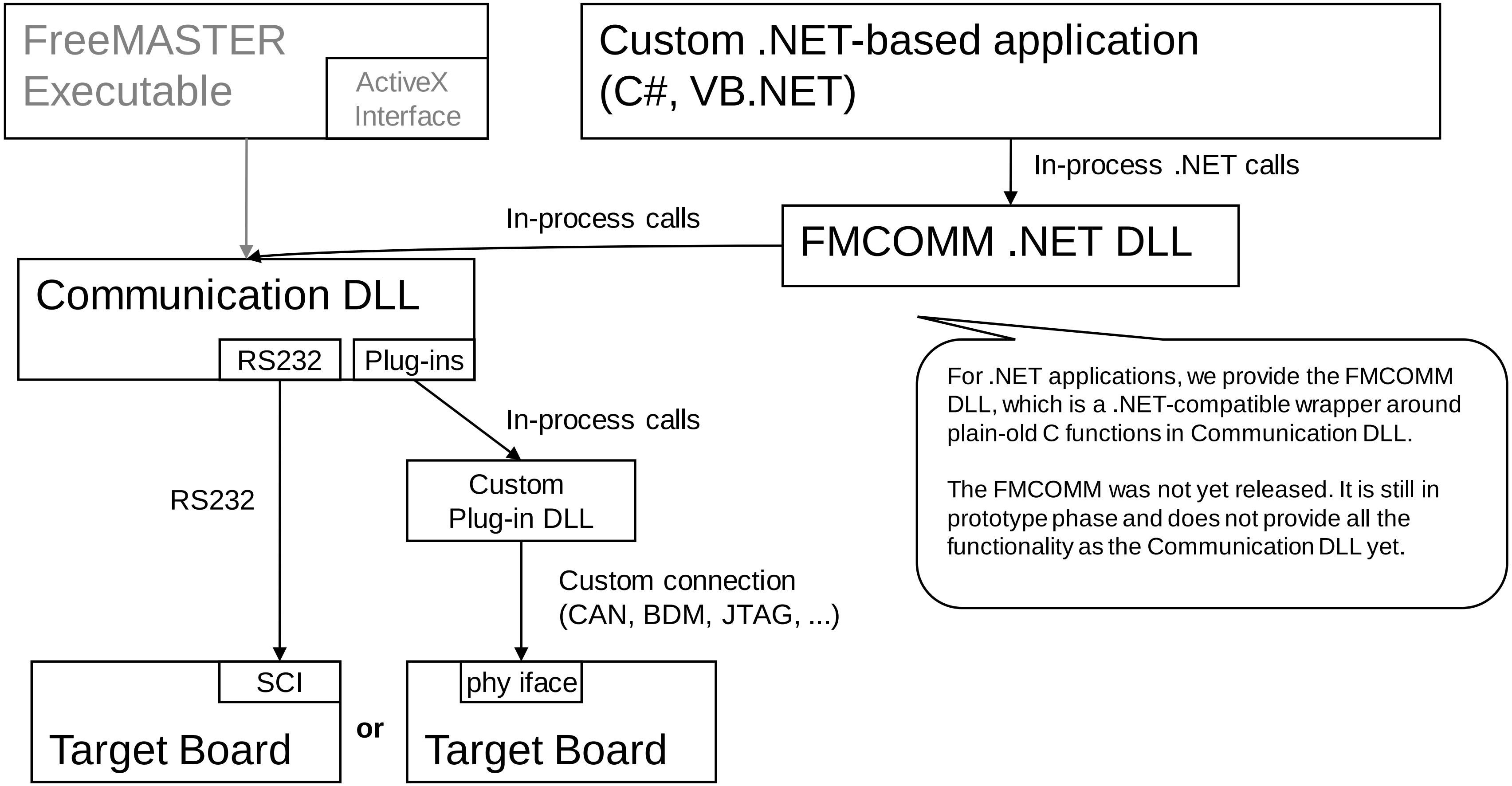
Any Windows-based application which is capable of direct C-like calls into the DLL may reuse the communication DLL to make use of the FreeMASTER protocol.

The DLL exports calls like OpenPort, ClosePort, ReadMemory, WriteMemory, SetupRecorder, ...

As FreeMASTER is out of the game here, the term “variable” makes no more sense in this scenario. Users application needs to use numeric memory addresses and sizes when accessing the board (see FM protocol for more details)

Internal Application Structure

- Other Ways to Access Target Microprocessor: .NET, C#, ...



For .NET applications, we provide the FMCOMM DLL, which is a .NET-compatible wrapper around plain-old C functions in Communication DLL.

The FMCOMM was not yet released. It is still in prototype phase and does not provide all the functionality as the Communication DLL yet.

Agenda

- FreeMASTER Overview
- **Quick_Start Overview**
- Processor Exert Overview
- Processor Expert Demo

What is Quick Start?

- ►Quick_Start = Easy-to-use Software Development Environment

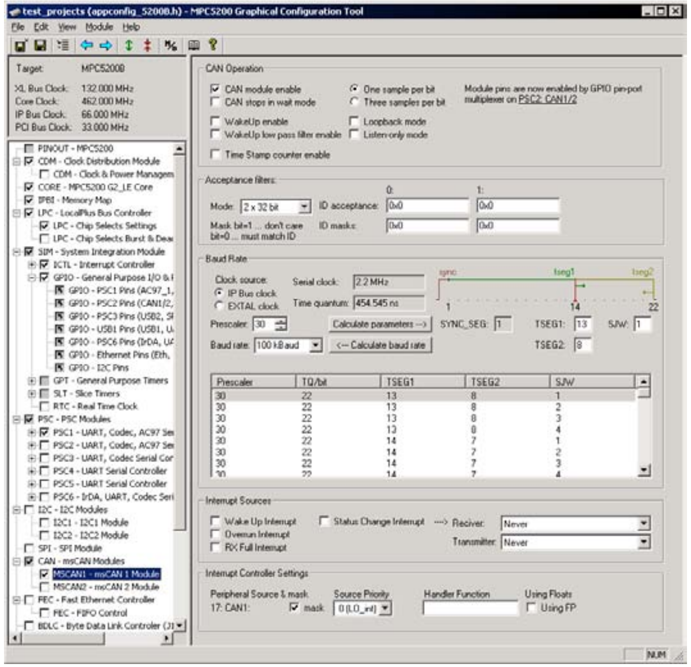
- ►Set of Low-level Drivers for all Peripheral Modules
  - C-language structures of peripheral memory space
  - Unified way of accessing peripheral registers
  - Highly-optimized to achieve an optimal assembly generated

- ►Ready-to-use Project Templates (“Project Stationery”)
  - Compiler configurations (RAM-debug, Flash-standalone targets)
  - Processor start-up code
  - Interrupt tables or Interrupt Dispatcher
  - Debugger initialization files

- ►Graphical Configuration Tool
  - User-friendly insight to processor configuration (cont.)

What is Quick Start?

- Graphical Configuration Tool
  - Edits post-reset processor configuration graphically
  - Configuration saved/read from a single ANSI C header file
  - GUI to configuration bits of all peripheral module registers
  - Possible conflict warnings
  - Pin-out view of processor I/O pins



test_projects (appconfig_5200B.h) - MPC5200 Graphical Configuration Tool

Target: MPC5200B
XL Bus Clock: 132.000 MHz
Core Clock: 462.000 MHz
IP Bus Clock: 66.000 MHz
PCI Bus Clock: 33.000 MHz

- ▣ PINOUT - MPC5200
- ☑ CDM - Clock Distribution Module
  - □ CDM - Clock & Power Managem
- ☑ CORE - MPC5200 G2_LE Core
- ☑ IPBI - Memory Map
- ☑ LPC - LocalPlus Bus Controller
  - ☑ LPC - Chip Selects Settings
  - □ LPC - Chip Selects Burst & Dea
- ☑ SIM - System Integration Module
  - ☑ ICTL - Interrupt Controller
  - ☑ GPIO - General Purpose I/O & I
    - ▣ GPIO - PSC1 Pins (AC97_1,
    - ▣ GPIO - PSC2 Pins (CAN1/2,
    - ▣ GPIO - PSC3 Pins (USB2, S
    - ▣ GPIO - USB1 Pins (USB1, U
    - ▣ GPIO - PSC6 Pins (IrDA, U
    - ▣ GPIO - Ethernet Pins (Eth,
    - ▣ GPIO - I2C Pins
  - ▣ GPT - General Purpose Timers
  - ▣ SLT - Slice Timers
  - □ RTC - Real Time Clock
- ☑ PSC - PSC Modules
  - ☑ PSC1 - UART, Codec, AC97 Se
  - □ PSC2 - UART, Codec, AC97 Se
  - □ PSC3 - UART, Codec Serial Co
  - □ PSC4 - UART Serial Controller
  - □ PSC5 - UART Serial Controller
  - □ PSC6 - IrDA, UART, Codec Seri
- □ I2C - I2C Modules
  - □ I2C1 - I2C1 Module
  - □ I2C2 - I2C2 Module
- □ SPI - SPI Module
- ☑ CAN - msCAN Modules
  - ☑ MSCAN1 - msCAN 1 Module
  - □ MSCAN2 - msCAN 2 Module
- □ FEC - Fast Ethernet Controller
  - □ FEC - FIFO Control
- □ BDLC - Byte Data Link Controler (J1

CAN Operation

☑ CAN module enable
□ CAN stops in wait mode
● One sample per bit
○ Three samples per bit
Module pins are now enabled by GPIO pin-port multiplexer on PSC2: CAN1/2
□ WakeUp enable
□ WakeUp low pass filter enable
□ Loopback mode
□ Listen-only mode
□ Time Stamp counter enable

Acceptance filters

Mode: 2 x 32 bit
ID acceptance: 0: 0x0 1: 0x0
ID masks: 0: 0x0 1: 0x0
Mask bit=1 ... don't care
bit=0 ... must match ID

Baud Rate

Clock source:
● IP Bus clock
○ EXTAL clock
Serial clock: 2.2 MHz
Time quantum: 454.545 ns
Prescaler: 30
Baud rate: 100 kBaud

SYNC_SEG: 1
TSEG1: 13
TSEG2: 8
SJW: 1

Prescaler	TQ/bit	TSEG1	TSEG2	SJW
30	22	13	8	1
30	22	13	8	2
30	22	13	8	3
30	22	13	8	4
30	22	14	7	1
30	22	14	7	2
30	22	14	7	3
30	22	14	7	4

Interrupt Sources

□ Wake Up Interrupt
□ Overrun Interrupt
□ RX Full Interrupt
□ Status Change Interrupt ---> Reciver: Never
Transmitter: Never

Interrupt Controller Settings

Peripheral Source & mask: 17: CAN1: ☑ mask
Source Priority: 0 (LO_int)
Handler Function:
Using Floats: □ Using FP

- Sample Applications
  - Demonstrating usage of GCT, processor peripheral modules and low-level drivers

- User Manual
  - Low-level drivers & tools guide
  - Latest device User Manual

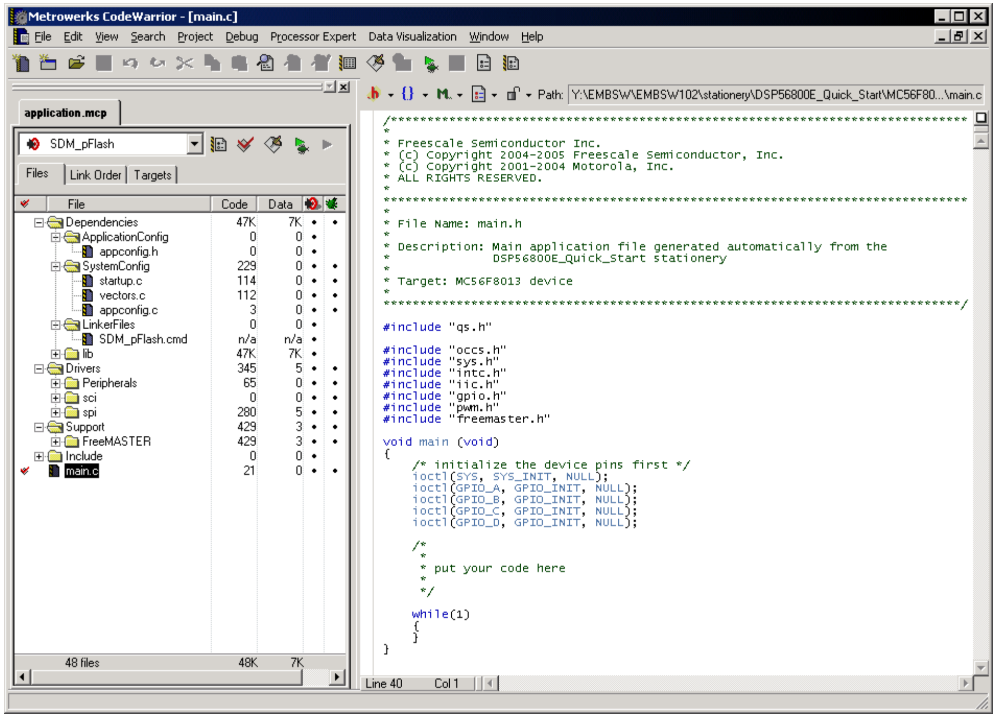
Quick Start Environment

►CodeWarrior Integration

- Quick_Start project stationery is installed directly into the CW
- Support for CW debugger and Flash Programmer
- GCT invoked from CW IDE

►Other Tools

- MPC500/MPC5500 supports makefile-based tools (Diab, Green Hills)
- Lauterbach Debugger



Metrowerks CodeWarrior - [main.c]
File Edit View Search Project Debug Processor Expert Data Visualization Window Help
Path: Y:\EMBSW\EMBSW102\stationery\DSP56800E_Quick_Start\MC56F80...\main.c
application.mcp
SDM_pFlash
Files Link Order Targets
Line 40 Col 1

File	Code	Data		
Dependencies	47K	7K	●	●
ApplicationConfig	0	0	●	
appconfig.h	0	0	●	
SystemConfig	229	0	●	●
startup.c	114	0	●	●
vectors.c	112	0	●	●
appconfig.c	3	0	●	●
LinkerFiles	0	0	●	
SDM_pFlash.cmd	n/a	n/a	●	
lib	47K	7K	●	
Drivers	345	5	●	●
Peripherals	65	0	●	●
sci	0	0	●	●
spi	280	5	●	●
Support	429	3	●	●
FreeMASTER	429	3	●	●
Include	0	0	●	
main.c	21	0	●	●
48 files	48K	7K		

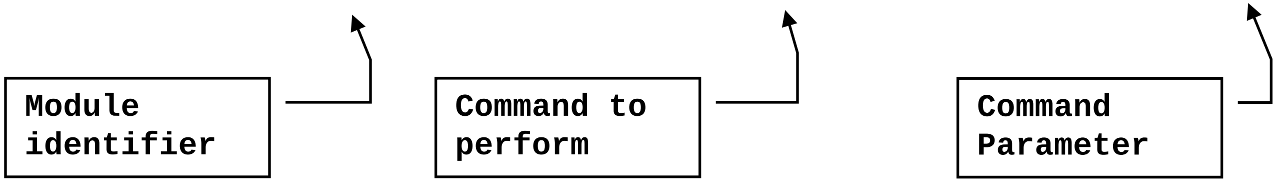
```
/*******************************************************************************
*
* Freescale Semiconductor Inc.
* (c) Copyright 2004-2005 Freescale Semiconductor, Inc.
* (c) Copyright 2001-2004 Motorola, Inc.
* ALL RIGHTS RESERVED.
*
********************************************************************************
*
* File Name: main.h
*
* Description: Main application file generated automatically from the
*              DSP56800E_Quick_Start stationery
*
* Target: MC56F8013 device
*
*******************************************************************************/

#include "qs.h"

#include "occs.h"
#include "sys.h"
#include "intc.h"
#include "iic.h"
#include "gpio.h"
#include "pwm.h"
#include "freemaster.h"

void main (void)
{
    /* initialize the device pins first */
    ioctl(SYS, SYS_INIT, NULL);
    ioctl(GPIO_A, GPIO_INIT, NULL);
    ioctl(GPIO_B, GPIO_INIT, NULL);
    ioctl(GPIO_C, GPIO_INIT, NULL);
    ioctl(GPIO_D, GPIO_INIT, NULL);

    /*
     *
     * put your code here
     *
     */

    while(1)
    {
    }
}
```

Low-level Drivers

▶ Quick Start Low-level Drivers

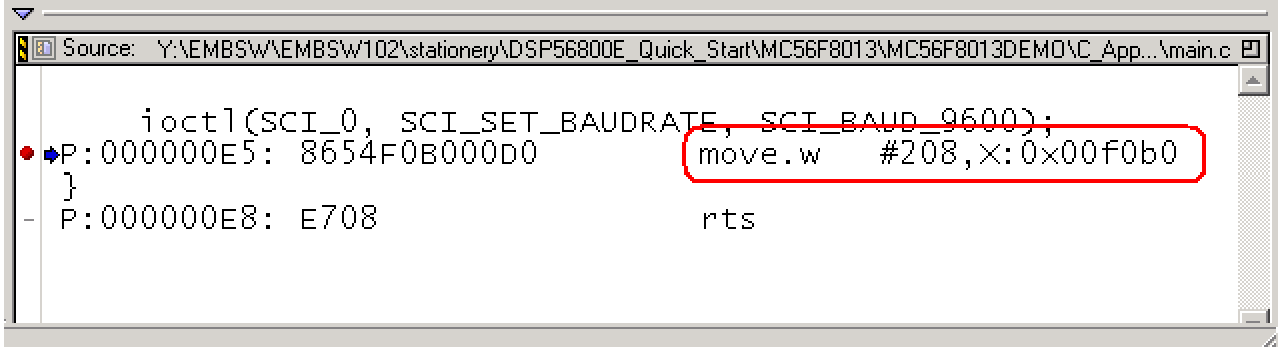
- Full control over and full access to all processor resources
- Unifies access to peripheral memory space (**`ioctl`** call)
- Registers are not accessed directly, although this is still possible
- **`ioctl`** calls are optimally compiled macros or functions

```
ioctl(SCI_0, SCI_SET_BAUDRATE, SCI_BAUD_9600)
```


Module identifier

Command to perform

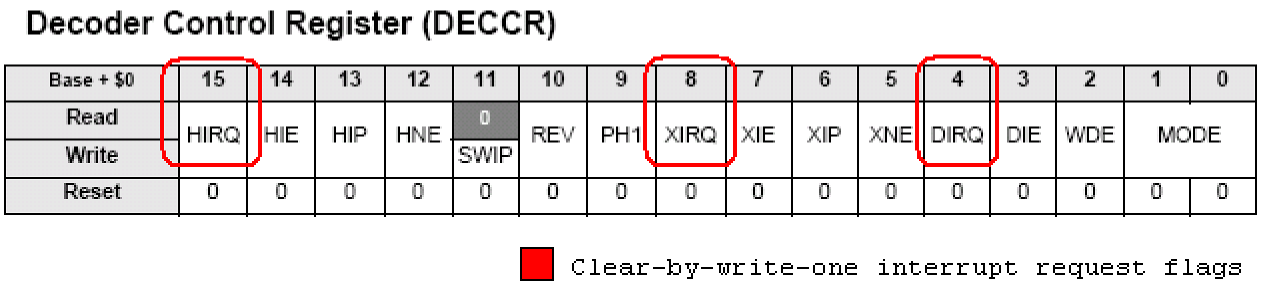
Command Parameter


Source: Y:\EMBSW\EMBSW102\stationery\DSP56800E_Quick_Start\MC56F8013\MC56F8013DEMO\C_App...\main.c

```
    ioctl(SCI_0, SCI_SET_BAUDRATE, SCI_BAUD_9600);
P:000000E5: 8654F0B000D0         move.w   #208,X:0x00f0b0
}
P:000000E8: E708                 rts
```

Low-level Drivers

- Why not to use direct access to peripheral registers?
  - Most of ioctl calls are “macroized” to direct register access anyway (either read/write or bit-set/bit-clear instructions used)
  - Some registers do need special attention, ioctl usage brings kind-of abstraction and transparency to an application code while still being optimally compiled



Decoder Control Register (DECCR)

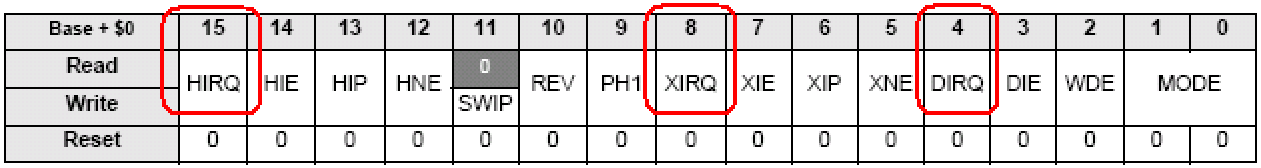
Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HIRQ	HIE	HIP	HNE	0	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	MODE	
Write					SWIP											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

🟥 Clear-by-write-one interrupt request flags

Exercise: Suppose you want to clear DIRQ bit only, while not modifying the rest of the register. Also you must not clear the HIRQ and XIRQ bits.
What C or assembly statement will you use on 56F800E?

solution on the next slide...

Decoder Control Register (DECCR)

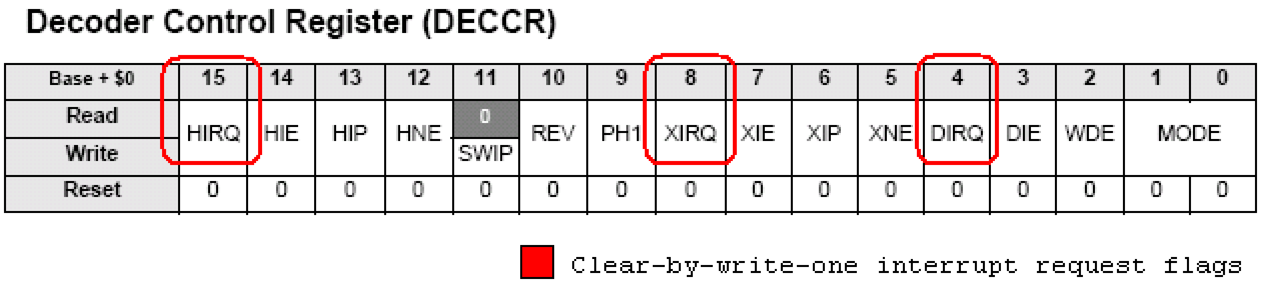


Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HIRQ	HIE	HIP	HNE	0	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	MODE	
Write					SWIP											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

🟥 Clear-by-write-one interrupt request flags

```
#define DECCR_DIRQ 0x0010  /* DIRQ bit constant */
ArchIO.Decoder0.deccr      /* register in the peripheral structure */
```

- DIRQ gets cleared ... OK
- XIRQ and HIRQ remain unchanged ... OK
- All other bits get reset! ... Wrong!



Decoder Control Register (DECCR)

Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HIRQ	HIE	HIP	HNE	0	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	MODE	
Write					SWIP											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

🟥 Clear-by-write-one interrupt request flags

```
#define DECCR_DIRQ 0x0010  /* DIRQ bit constant */
ArchIO.Decoder0.deccr      /* register in the peripheral structure */
```

C-language:

```
    ArchIO.Decoder0.deccr = DECCR_DIRQ;
```

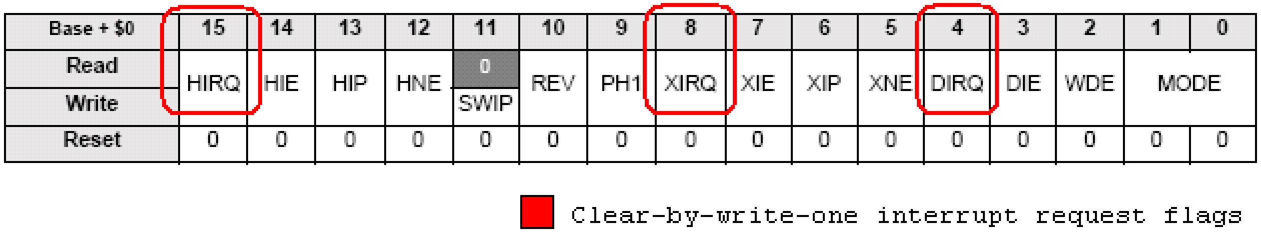
56F800E Assembler:

```
    asm ( move.w #>16,X:0x00f180 );
```

- DIRQ gets cleared ... OK
- XIRQ and HIRQ remain unchanged ... OK
- **All other bits get reset! ... Wrong!**

Low-level Drivers: Exercise

Decoder Control Register (DECCR)



Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HIRQ	HIE	HIP	HNE	0	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	MODE	
Write					SWIP											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

🟥 Clear-by-write-one interrupt request flags

```
#define DECCR_DIRQ 0x0010  /* DIRQ bit constant */
ArchIO.Decoder0.deccr      /* register in the peripheral structure */
```

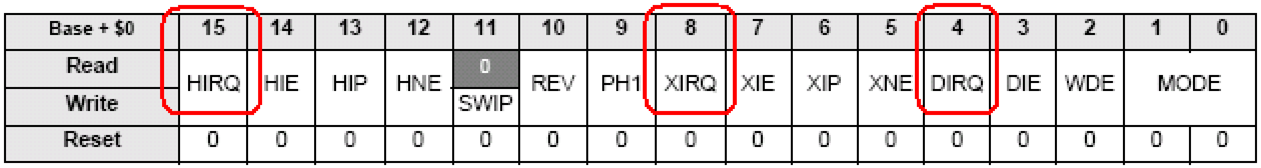
```
C-language:
    ArchIO.Decoder0.deccr |= DECCR_DIRQ;
```

```
56F800E Assembler:
    asm ( bfset #0x10,X:0x00f180 );
```

- DIRQ gets cleared ... OK
- Other register bits unchanged ... OK
- XIRQ or HIRQ gets reset if they read as “1”
(i.e. when interrupt request is pending!)

Low-level Drivers: Exercise

Decoder Control Register (DECCR)



Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HIRQ	HIE	HIP	HNE	0	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	MODE	
Write					SWIP											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

🟥 Clear-by-write-one interrupt request flags

```
#define DECCR_DIRQ 0x0010  /* DIRQ bit constant */
#define DECCR_HIRQ 0x8000  /* HIRQ bit constant */
#define DECCR_XIRQ 0x0100  /* XIRQ bit constant */
ArchIO.Decoder0.deccr      /* register in the peripheral structure */
```

C-language:

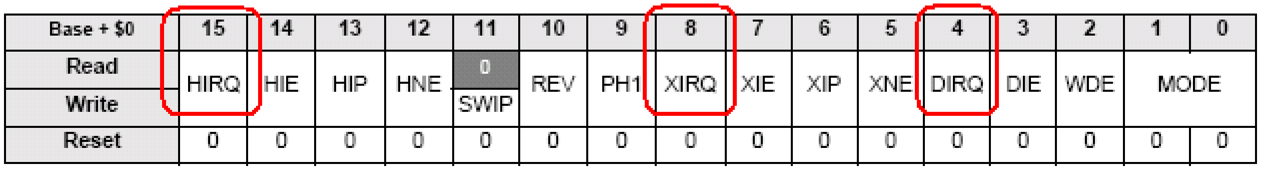
```
    ArchIO.Decoder0.deccr &= ~(~(DECCR_DIRQ) &
       (DECCR_HIRQ | DECCR_XIRQ));
```

56F800E Assembler:

```
    asm ( bfclr #0x8100,X:0x00f180 );
```

Low-level Drivers: Exercise

Decoder Control Register (DECCR)



Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HIRQ	HIE	HIP	HNE	0	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	MODE	
Write					SWIP											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

🟥 Clear-by-write-one interrupt request flags

```
#define DECCR_DIRQ 0x0010  /* DIRQ bit constant */
#define DECCR_HIRQ 0x8000  /* HIRQ bit constant */
#define DECCR_XIRQ 0x0100  /* XIRQ bit constant */
ArchIO.Decoder0.deccr      /* register in the peripheral structure */
```

```
C-language:
    ArchIO.Decoder0.deccr &= ~(~(DECCR_DIRQ) &
        (DECCR_HIRQ | DECCR_XIRQ));
```

```
56F800E Assembler:
    asm ( bfclr #0x8100,X:0x00f180 );
```

Better work with Quick_Start and use the“Clear Interrupt Request” command:

ioctl(DEC_0, DEC_INT_REQUEST_CLEAR, DEC_DECCR_DIRQ);

Low-level Drivers: Highlights

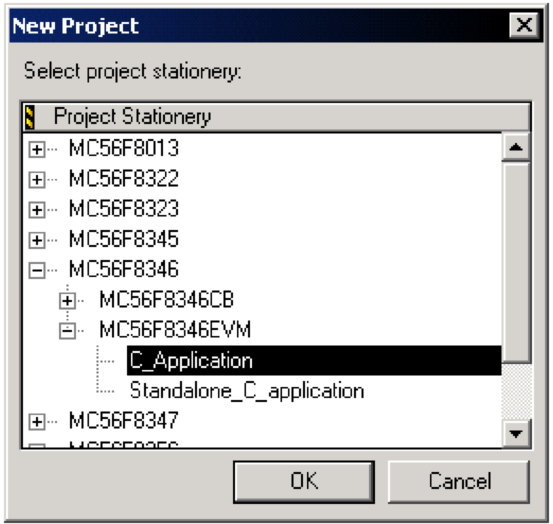
- Low-level Drivers Highlights
  - Full control over all processor resources
  - Real-world application development know-how inside
    - transparent solution to tricky register access
    - higher abstraction and code readability without loosing performance
  - Delivered as source code
  - Fully tested and documented

Project Stationery

Project Stationery

- Quick_Start Project Stationery

- CodeWarrior concept of creating a new project
  - CodeWarrior “clones” the project template and creates a ready-to-use skeleton of a new application
  - In Quick_Start, a dedicated project stationery exists for each processor and evaluation board (EVB)
    - Processors differ in memory layout, peripheral modules etc.
    - For a given processor, more than one EVB may exist, differing in how the processor is connected with external components



New Project

Select project stationery:

```
Project Stationery
+ MC56F8013
+ MC56F8322
+ MC56F8323
+ MC56F8345
- MC56F8346
    + MC56F8346CB
    - MC56F8346EVM
        C_Application
        Standalone_C_application
+ MC56F8347
```

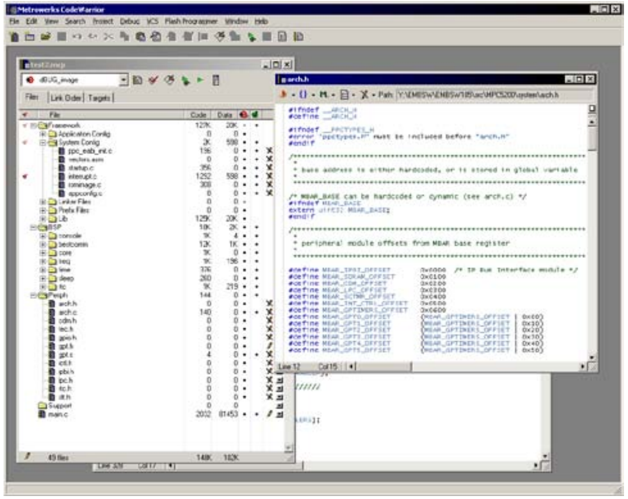
Project Stationery

- Quick_Start Project Stationery
  - Multiple Compiler configurations per project
    - RAM-based debugging targets
    - Standalone Flash-based (release) targets
    - CPU Simulator target
  - Start-up code, Board Initialization, Interrupt tables
  - Linker Command Files
    - Provide the linker with information about how to arrange a C-code in memory
  - Debugger Configuration Files
    - Making the EVB ready for RAM-based debugging
    - Making the EVB ready for Flash Programmer
    - Memory description files

Configuration

Graphical Configuration Tool

- Graphical Configuration Tool (GCT)
- A desktop application for MS Windows XP (W2000, NT)
  - Used to edit the ANSI C-compatible application configuration header file (typically appconfig.h for Quick_Start applications)

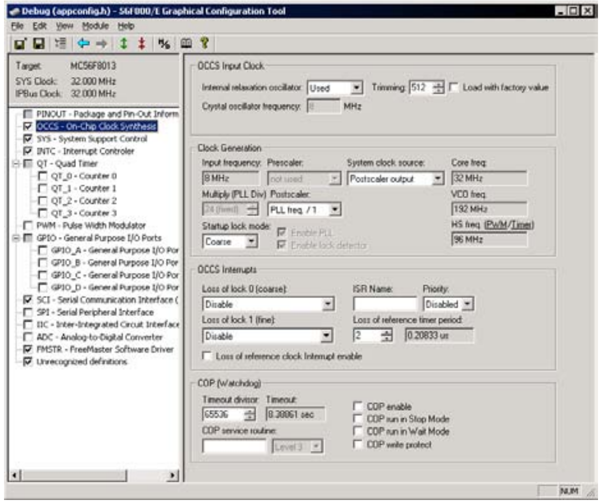


Metrowerks CodeWarrior
File Edit View Search Project Debug VCS Flash Programmer Window Help
arch.h
Path: Y:\EMBSW\EMBSW185\src\MPC5200\system\arch.h
Line 12 Col 15

```
#ifndef __ARCH_H
#define __ARCH_H

#ifndef __PPCTYPES_H
#error "ppctypes.h" must be included before "arch.h"
#endif

/*************************************************************************
 *
 * base address is either hardcoded, or is stored in global variable
 *
 *************************************************************************

/* MBAR_BASE can be hardcoded or dynamic (see arch.c) */
#ifndef MBAR_BASE
extern uint32 MBAR_BASE;
#endif

/*************************************************************************
 *
 * peripheral module offsets from MBAR base register
 *
 *************************************************************************

#define MBAR_IPBI_OFFSET        0x0000  /* IP Bus Interface module */
#define MBAR_SDRAM_OFFSET       0x0100
#define MBAR_CDM_OFFSET         0x0200
#define MBAR_LPC_OFFSET         0x0300
#define MBAR_SCTMR_OFFSET       0x0400
#define MBAR_INT_CTRL_OFFSET    0x0500
#define MBAR_GPTIMERS_OFFSET    0x0600
#define MBAR_GPT0_OFFSET        (MBAR_GPTIMERS_OFFSET | 0x00)
#define MBAR_GPT1_OFFSET        (MBAR_GPTIMERS_OFFSET | 0x10)
#define MBAR_GPT2_OFFSET        (MBAR_GPTIMERS_OFFSET | 0x20)
#define MBAR_GPT3_OFFSET        (MBAR_GPTIMERS_OFFSET | 0x30)
#define MBAR_GPT4_OFFSET        (MBAR_GPTIMERS_OFFSET | 0x40)
#define MBAR_GPT5_OFFSET        (MBAR_GPTIMERS_OFFSET | 0x50)
```

Metrowerks CodeWarrior IDE

Ctrl+F10 invoked GCT opens the appconfig.h for a current project



Debug (appconfig.h) - 56F800/E Graphical Configuration Tool
File Edit View Module Help
NUM

Target: MC56F8013
SYS Clock: 32.000 MHz
IPBus Clock: 32.000 MHz

- □ PINOUT - Package and Pin-Out Inform
- ☑ OCCS - On-Chip Clock Synthesis
- ☑ SYS - System Support Control
- ☑ INTC - Interrupt Controler
- □ QT - Quad Timer
  - □ QT_0 - Counter 0
  - □ QT_1 - Counter 1
  - □ QT_2 - Counter 2
  - □ QT_3 - Counter 3
- □ PWM - Pulse Width Modulator
- □ GPIO - General Purpose I/O Ports
  - □ GPIO_A - General Purpose I/O Por
  - □ GPIO_B - General Purpose I/O Por
  - □ GPIO_C - General Purpose I/O Por
  - □ GPIO_D - General Purpose I/O Por
- ☑ SCI - Serial Communication Interface (
- □ SPI - Serial Peripheral Interface
- □ IIC - Inter-Integrated Circuit Interface
- □ ADC - Analog-to-Digital Converter
- ☑ FMSTR - FreeMaster Software Driver
- ☑ Unrecognized definitions

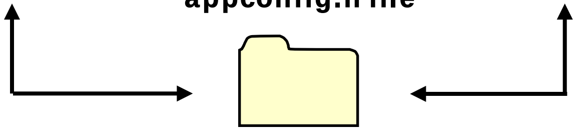
OCCS Input Clock
Internal relaxation oscillator: Used Trimming: 512 □ Load with factory value
Crystal oscillator frequency: 8 MHz

Clock Generation
Input frequency: 8 MHz Prescaler: not used System clock source: Postscaler output Core freq: 32 MHz
Multiply (PLL Div): 24 (fixed) Postscaler: PLL freq. / 1 VCO freq: 192 MHz
Startup lock mode: Coarse ☑ Enable PLL ☑ Enable lock detector HS freq. (PWM/Timer): 96 MHz

OCCS Interrupts
Loss of lock 0 (coarse): Disable ISR Name: Priority: Disabled
Loss of lock 1 (fine): Disable Loss of reference timer period: 2 0.20833 us
□ Loss of reference clock Interrupt enable

COP (Watchdog)
Timeout divisor: 65536 Timeout: 0.20061 sec
COP service routine: Level 3
□ COP enable
□ COP run in Stop Mode
□ COP run in Wait Mode
□ COP write protect

Graphical Configuration Tool



appconfig.h file

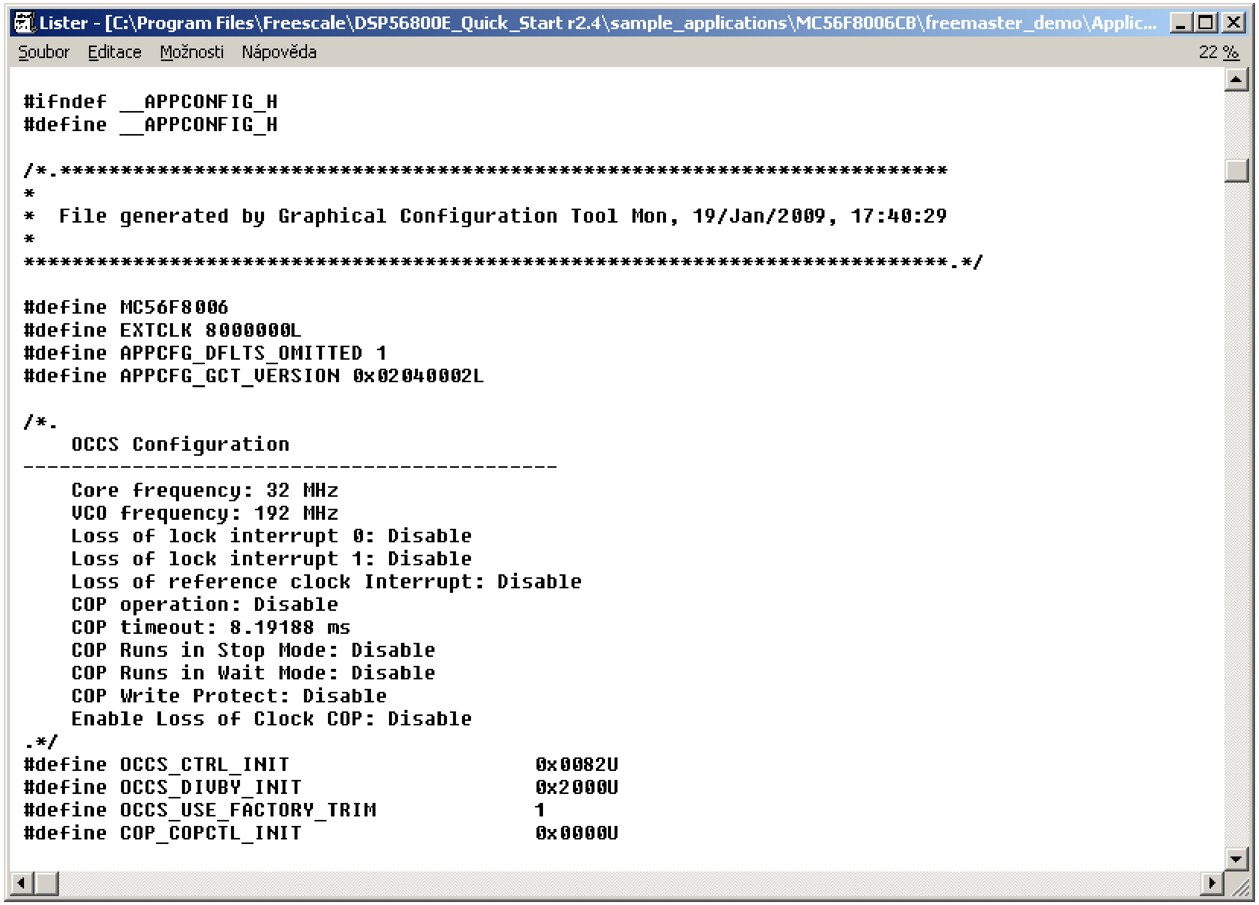
#include “appconfig.h”
#defines used to initialize peripherals

Read & Write access to appconfig.h

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © Freescale Semiconductor, Inc. 2009.

Graphical Configuration Tool: appconfig.h

- Configuration File Example


Lister - [C:\Program Files\Freescale\DSP56800E_Quick_Start r2.4\sample_applications\MC56F8006CB\freemaster_demo\Applic...
Soubor
Editace
Možnosti
Nápověda
22 %

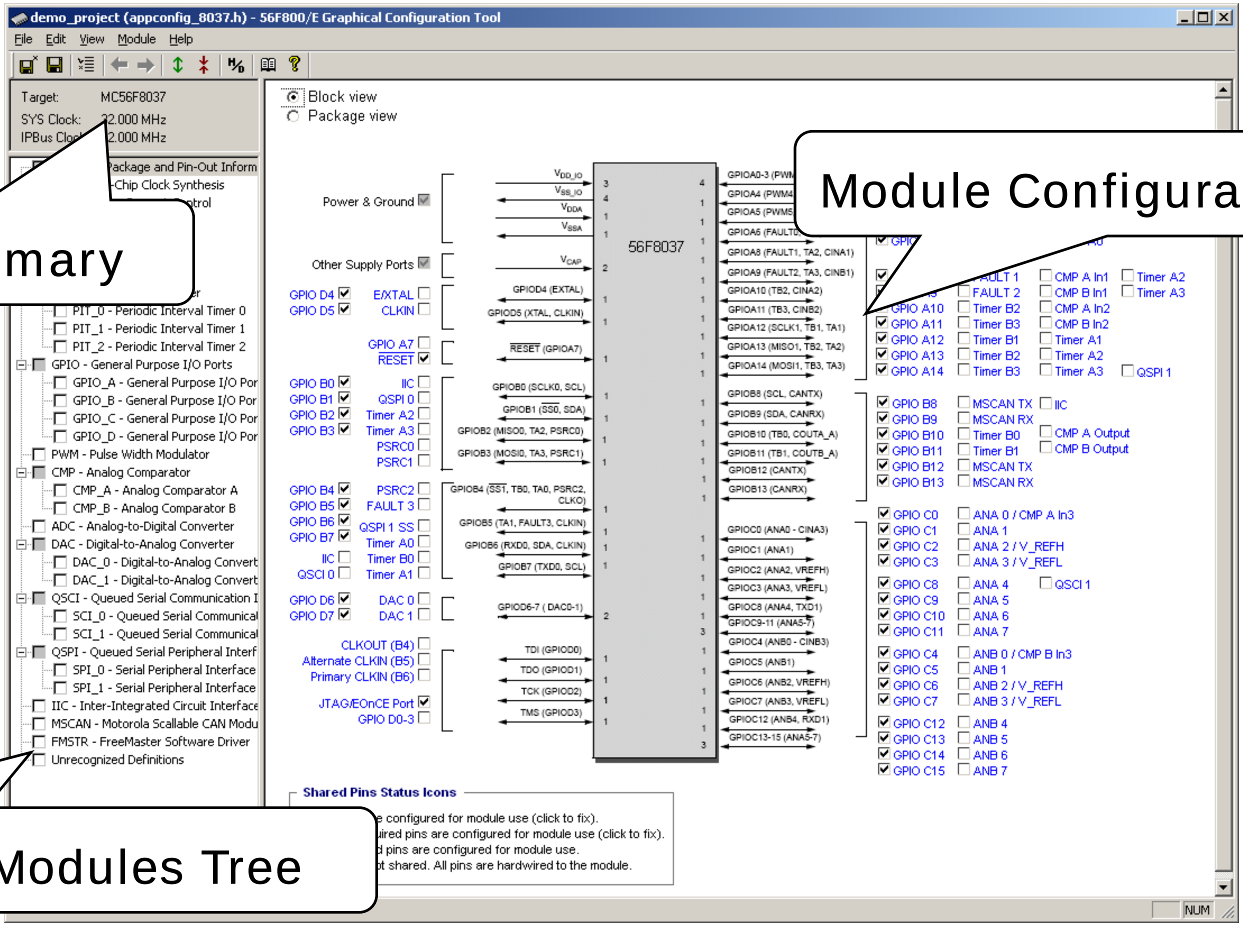
```
#ifndef __APPCONFIG_H
#define __APPCONFIG_H

/*.**************************************************************************
*
*  File generated by Graphical Configuration Tool Mon, 19/Jan/2009, 17:40:29
*
*****************************************************************************.*/

#define MC56F8006
#define EXTCLK 8000000L
#define APPCFG_DFLTS_OMITTED 1
#define APPCFG_GCT_VERSION 0x02040002L

/*.
    OCCS Configuration
--------------------------------------------
    Core frequency: 32 MHz
    VCO frequency: 192 MHz
    Loss of lock interrupt 0: Disable
    Loss of lock interrupt 1: Disable
    Loss of reference clock Interrupt: Disable
    COP operation: Disable
    COP timeout: 8.19188 ms
    COP Runs in Stop Mode: Disable
    COP Runs in Wait Mode: Disable
    COP Write Protect: Disable
    Enable Loss of Clock COP: Disable
.*/
#define OCCS_CTRL_INIT                    0x0082U
#define OCCS_DIVBY_INIT                   0x2000U
#define OCCS_USE_FACTORY_TRIM             1
#define COP_COPCTL_INIT                   0x0000U
```

Graphical Configuration Tool: appconfig.h

- GCT and the “appconfig.h” File
- A single macro constant per peripheral register
- Configuration summary comments
- Read / Write in GCT
  - Enables manual editing of the appconfig.h file
  - Copy & paste migrating to other CPUs
  - GCT supports importing of module configuration within a single project or between projects
- Private section in appconfig.h file
  - Users put other global symbols & definitions here
  - The file can be a real application configuration file (not only the processor configuration)

Graphical Configuration Tool

- Different Control Page for each Peripheral Module



demo_project (appconfig_8037.h) - 56F800/E Graphical Configuration Tool

Target: MC56F8037
SYS Clock: 32.000 MHz
IPBus Clock: 32.000 MHz

- □ Package and Pin-Out Inform
- □ On-Chip Clock Synthesis
- □ ... Control
- □ PIT_0 - Periodic Interval Timer 0
- □ PIT_1 - Periodic Interval Timer 1
- □ PIT_2 - Periodic Interval Timer 2
- □ GPIO - General Purpose I/O Ports
  - □ GPIO_A - General Purpose I/O Por
  - □ GPIO_B - General Purpose I/O Por
  - □ GPIO_C - General Purpose I/O Por
  - □ GPIO_D - General Purpose I/O Por
- □ PWM - Pulse Width Modulator
- □ CMP - Analog Comparator
  - □ CMP_A - Analog Comparator A
  - □ CMP_B - Analog Comparator B
- □ ADC - Analog-to-Digital Converter
- □ DAC - Digital-to-Analog Converter
  - □ DAC_0 - Digital-to-Analog Convert
  - □ DAC_1 - Digital-to-Analog Convert
- □ QSCI - Queued Serial Communication I
  - □ SCI_0 - Queued Serial Communical
  - □ SCI_1 - Queued Serial Communical
- □ QSPI - Queued Serial Peripheral Interf
  - □ SPI_0 - Serial Peripheral Interface
  - □ SPI_1 - Serial Peripheral Interface
- □ IIC - Inter-Integrated Circuit Interface
- □ MSCAN - Motorola Scalable CAN Modu
- □ FMSTR - FreeMaster Software Driver
- □ Unrecognized Definitions

● Block view
○ Package view

Power & Ground ☑
Other Supply Ports ☑
GPIO D4 ☑ E/XTAL □
GPIO D5 ☑ CLKIN □
GPIO A7 □
RESET ☑
GPIO B0 ☑ IIC □
GPIO B1 ☑ QSPI 0 □
GPIO B2 ☑ Timer A2 □
GPIO B3 ☑ Timer A3 □
PSRC0 □
PSRC1 □
GPIO B4 ☑ PSRC2 □
GPIO B5 ☑ FAULT 3 □
GPIO B6 ☑ QSPI 1 SS □
GPIO B7 ☑ Timer A0 □
IIC □ Timer B0 □
QSCI 0 □ Timer A1 □
GPIO D6 ☑ DAC 0 □
GPIO D7 ☑ DAC 1 □
CLKOUT (B4) □
Alternate CLKIN (B5) □
Primary CLKIN (B6) □
JTAG/EOnCE Port ☑
GPIO D0-3 □

FAULT 1 □ CMP A In1 □ Timer A2
FAULT 2 □ CMP B In1 □ Timer A3
GPIO A10 □ Timer B2 □ CMP A In2
☑ GPIO A11 □ Timer B3 □ CMP B In2
☑ GPIO A12 □ Timer B1 □ Timer A1
☑ GPIO A13 □ Timer B2 □ Timer A2
☑ GPIO A14 □ Timer B3 □ Timer A3 □ QSPI 1
☑ GPIO B8 □ MSCAN TX □ IIC
☑ GPIO B9 □ MSCAN RX
☑ GPIO B10 □ Timer B0 □ CMP A Output
☑ GPIO B11 □ Timer B1 □ CMP B Output
☑ GPIO B12 □ MSCAN TX
☑ GPIO B13 □ MSCAN RX
☑ GPIO C0 □ ANA 0 / CMP A In3
☑ GPIO C1 □ ANA 1
☑ GPIO C2 □ ANA 2 / V_REFH
☑ GPIO C3 □ ANA 3 / V_REFL
☑ GPIO C8 □ ANA 4 □ QSCI 1
☑ GPIO C9 □ ANA 5
☑ GPIO C10 □ ANA 6
☑ GPIO C11 □ ANA 7
☑ GPIO C4 □ ANB 0 / CMP B In3
☑ GPIO C5 □ ANB 1
☑ GPIO C6 □ ANB 2 / V_REFH
☑ GPIO C7 □ ANB 3 / V_REFL
☑ GPIO C12 □ ANB 4
☑ GPIO C13 □ ANB 5
☑ GPIO C14 □ ANB 6
☑ GPIO C15 □ ANB 7

Shared Pins Status Icons
...e configured for module use (click to fix).
...uired pins are configured for module use (click to fix).
...d pins are configured for module use.
...ot shared. All pins are hardwired to the module.

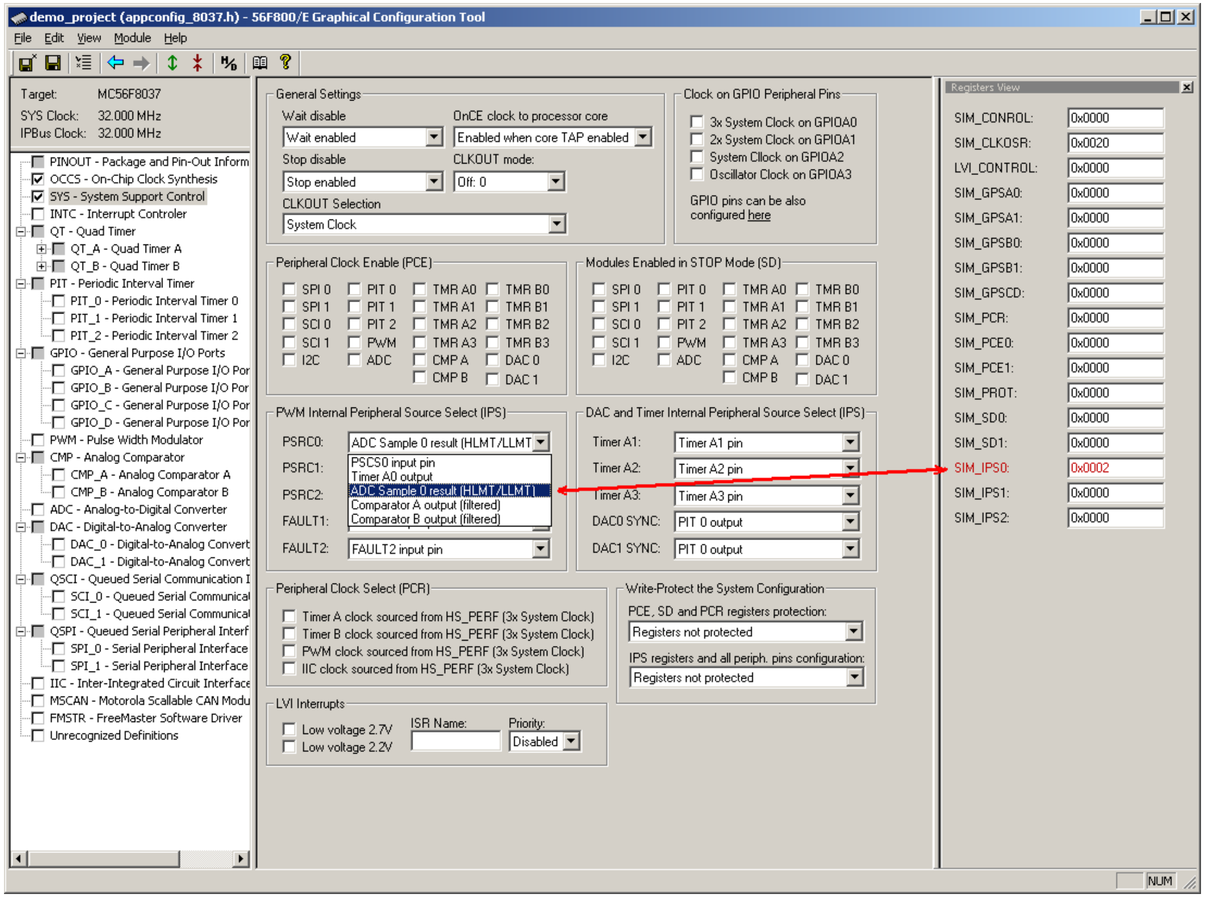
Module Configuration Page

Clocks Summary

Peripheral Modules Tree

Graphical Configuration Tool

- Direct Register Value View (two-way editing)



demo_project (appconfig_8037.h) - 56F800/E Graphical Configuration Tool

Target: MC56F8037
SYS Clock: 32.000 MHz
IPBus Clock: 32.000 MHz

- □ PINOUT - Package and Pin-Out Inform
- ☑ OCCS - On-Chip Clock Synthesis
- ☑ SYS - System Support Control
- □ INTC - Interrupt Controler
- □ QT - Quad Timer
  - □ QT_A - Quad Timer A
  - □ QT_B - Quad Timer B
- □ PIT - Periodic Interval Timer
  - □ PIT_0 - Periodic Interval Timer 0
  - □ PIT_1 - Periodic Interval Timer 1
  - □ PIT_2 - Periodic Interval Timer 2
- □ GPIO - General Purpose I/O Ports
  - □ GPIO_A - General Purpose I/O Por
  - □ GPIO_B - General Purpose I/O Por
  - □ GPIO_C - General Purpose I/O Por
  - □ GPIO_D - General Purpose I/O Por
- □ PWM - Pulse Width Modulator
- □ CMP - Analog Comparator
  - □ CMP_A - Analog Comparator A
  - □ CMP_B - Analog Comparator B
- □ ADC - Analog-to-Digital Converter
- □ DAC - Digital-to-Analog Converter
  - □ DAC_0 - Digital-to-Analog Convert
  - □ DAC_1 - Digital-to-Analog Convert
- □ QSCI - Queued Serial Communication I
  - □ SCI_0 - Queued Serial Communical
  - □ SCI_1 - Queued Serial Communical
- □ QSPI - Queued Serial Peripheral Interf
  - □ SPI_0 - Serial Peripheral Interface
  - □ SPI_1 - Serial Peripheral Interface
- □ IIC - Inter-Integrated Circuit Interface
- □ MSCAN - Motorola Scallable CAN Modu
- □ FMSTR - FreeMaster Software Driver
- □ Unrecognized Definitions

General Settings

Wait disable: Wait enabled
OnCE clock to processor core: Enabled when core TAP enabled
Stop disable: Stop enabled
CLKOUT mode: Off: 0
CLKOUT Selection: System Clock

Clock on GPIO Peripheral Pins

□ 3x System Clock on GPIOA0
□ 2x System Clock on GPIOA1
□ System Clock on GPIOA2
□ Oscillator Clock on GPIOA3

GPIO pins can be also configured here

Peripheral Clock Enable (PCE)

□ SPI 0	□ PIT 0	□ TMR A0	□ TMR B0
□ SPI 1	□ PIT 1	□ TMR A1	□ TMR B1
□ SCI 0	□ PIT 2	□ TMR A2	□ TMR B2
□ SCI 1	□ PWM	□ TMR A3	□ TMR B3
□ I2C	□ ADC	□ CMP A	□ DAC 0
		□ CMP B	□ DAC 1

Modules Enabled in STOP Mode (SD)

□ SPI 0	□ PIT 0	□ TMR A0	□ TMR B0
□ SPI 1	□ PIT 1	□ TMR A1	□ TMR B1
□ SCI 0	□ PIT 2	□ TMR A2	□ TMR B2
□ SCI 1	□ PWM	□ TMR A3	□ TMR B3
□ I2C	□ ADC	□ CMP A	□ DAC 0
		□ CMP B	□ DAC 1

PWM Internal Peripheral Source Select (IPS)

PSRC0: ADC Sample 0 result (HLMT/LLMT)
PSRC1:
PSRC2:
FAULT1:
FAULT2: FAULT2 input pin

PSCS0 input pin
Timer A0 output
ADC Sample 0 result (HLMT/LLMT)
Comparator A output (filtered)
Comparator B output (filtered)

DAC and Timer Internal Peripheral Source Select (IPS)

Timer A1: Timer A1 pin
Timer A2: Timer A2 pin
Timer A3: Timer A3 pin
DAC0 SYNC: PIT 0 output
DAC1 SYNC: PIT 0 output

Peripheral Clock Select (PCR)

□ Timer A clock sourced from HS_PERF (3x System Clock)
□ Timer B clock sourced from HS_PERF (3x System Clock)
□ PWM clock sourced from HS_PERF (3x System Clock)
□ IIC clock sourced from HS_PERF (3x System Clock)

Write-Protect the System Configuration

PCE, SD and PCR registers protection: Registers not protected
IPS registers and all periph. pins configuration: Registers not protected

LVI Interrupts

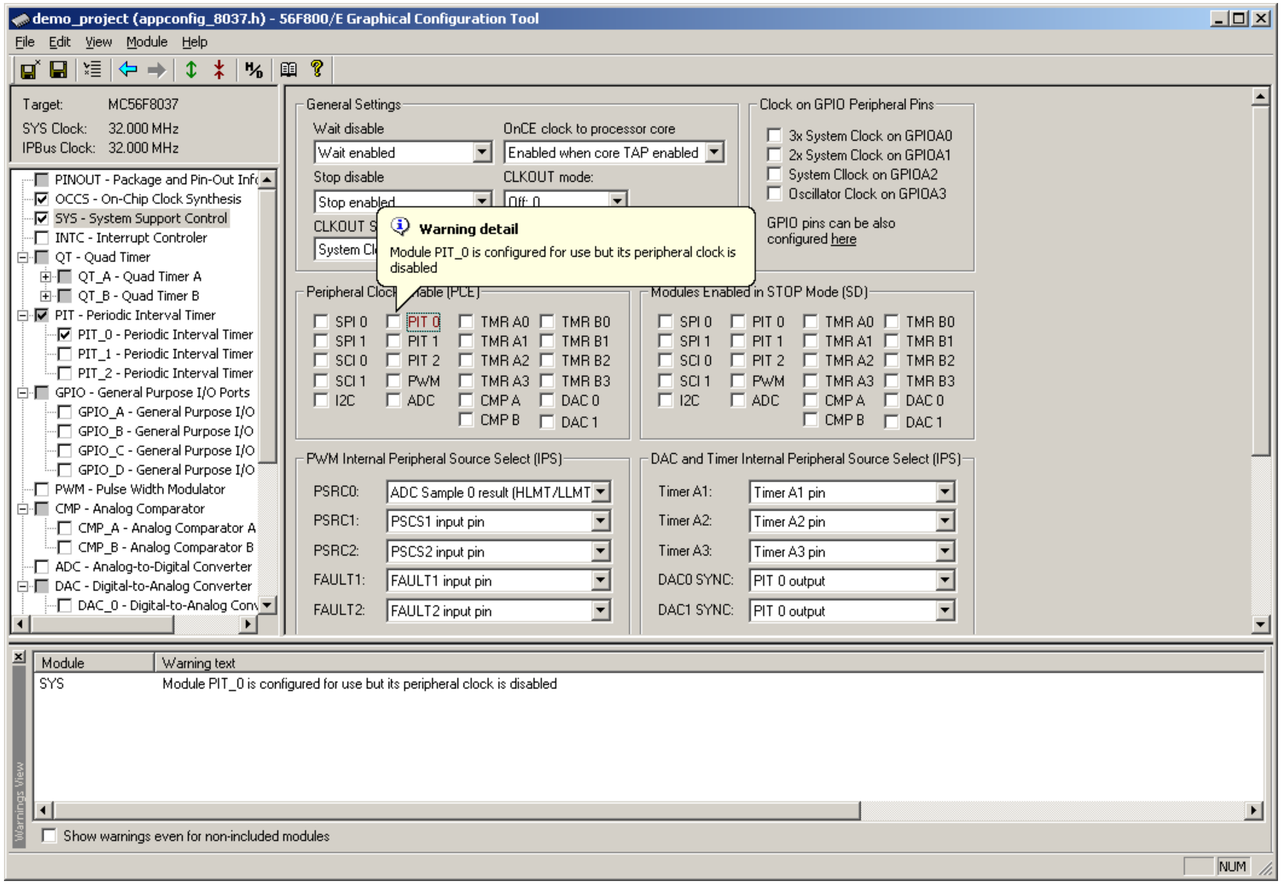
□ Low voltage 2.7V
□ Low voltage 2.2V
ISR Name:
Priority: Disabled

Registers View

SIM_CONROL:	0x0000
SIM_CLKOSR:	0x0020
LVI_CONTROL:	0x0000
SIM_GPSA0:	0x0000
SIM_GPSA1:	0x0000
SIM_GPSB0:	0x0000
SIM_GPSB1:	0x0000
SIM_GPSCD:	0x0000
SIM_PCR:	0x0000
SIM_PCE0:	0x0000
SIM_PCE1:	0x0000
SIM_PROT:	0x0000
SIM_SD0:	0x0000
SIM_SD1:	0x0000
SIM_IPS0:	0x0002
SIM_IPS1:	0x0000
SIM_IPS2:	0x0000

Graphical Configuration Tool

▶ Configuration Conflict Warnings


demo_project (appconfig_8037.h) - 56F800/E Graphical Configuration Tool
File
Edit
View
Module
Help

Target: MC56F8037
SYS Clock: 32.000 MHz
IPBus Clock: 32.000 MHz

- □ PINOUT - Package and Pin-Out Info
- ☑ OCCS - On-Chip Clock Synthesis
- ☑ SYS - System Support Control
- □ INTC - Interrupt Controler
- □ QT - Quad Timer
  - □ QT_A - Quad Timer A
  - □ QT_B - Quad Timer B
- ☑ PIT - Periodic Interval Timer
  - ☑ PIT_0 - Periodic Interval Timer
  - □ PIT_1 - Periodic Interval Timer
  - □ PIT_2 - Periodic Interval Timer
- □ GPIO - General Purpose I/O Ports
  - □ GPIO_A - General Purpose I/O
  - □ GPIO_B - General Purpose I/O
  - □ GPIO_C - General Purpose I/O
  - □ GPIO_D - General Purpose I/O
- □ PWM - Pulse Width Modulator
- □ CMP - Analog Comparator
  - □ CMP_A - Analog Comparator A
  - □ CMP_B - Analog Comparator B
- □ ADC - Analog-to-Digital Converter
- □ DAC - Digital-to-Analog Converter
  - □ DAC_0 - Digital-to-Analog Conv

General Settings

Wait disable: Wait enabled
OnCE clock to processor core: Enabled when core TAP enabled
Stop disable: Stop enabled
CLKOUT mode: Off: 0
CLKOUT S
System Cl

Clock on GPIO Peripheral Pins

□ 3x System Clock on GPIOA0
□ 2x System Clock on GPIOA1
□ System Cllock on GPIOA2
□ Oscillator Clock on GPIOA3

GPIO pins can be also configured here

Warning detail

Module PIT_0 is configured for use but its peripheral clock is disabled

Peripheral Clock Enable (PCE)

□ SPI 0 □ PIT 0 □ TMR A0 □ TMR B0
□ SPI 1 □ PIT 1 □ TMR A1 □ TMR B1
□ SCI 0 □ PIT 2 □ TMR A2 □ TMR B2
□ SCI 1 □ PWM □ TMR A3 □ TMR B3
□ I2C □ ADC □ CMP A □ DAC 0
□ CMP B □ DAC 1

Modules Enabled in STOP Mode (SD)

□ SPI 0 □ PIT 0 □ TMR A0 □ TMR B0
□ SPI 1 □ PIT 1 □ TMR A1 □ TMR B1
□ SCI 0 □ PIT 2 □ TMR A2 □ TMR B2
□ SCI 1 □ PWM □ TMR A3 □ TMR B3
□ I2C □ ADC □ CMP A □ DAC 0
□ CMP B □ DAC 1

PWM Internal Peripheral Source Select (IPS)

PSRC0: ADC Sample 0 result (HLMT/LLMT
PSRC1: PSCS1 input pin
PSRC2: PSCS2 input pin
FAULT1: FAULT1 input pin
FAULT2: FAULT2 input pin

DAC and Timer Internal Peripheral Source Select (IPS)

Timer A1: Timer A1 pin
Timer A2: Timer A2 pin
Timer A3: Timer A3 pin
DAC0 SYNC: PIT 0 output
DAC1 SYNC: PIT 0 output

Warnings View

Module	Warning text
SYS	Module PIT_0 is configured for use but its peripheral clock is disabled

□ Show warnings even for non-included modules

Agenda

- FreeMASTER Overview
- Quick_Start Overview
- **Processor Exert Overview**
- Processor Expert Demo

What is Processor Expert?

A rapid application design tool with …

- **Graphical User Interface** which allows an application to be specified by the functionality needed
- **Automatic code generator** which creates tested, optimized C code tuned to the application needs and selected Freescale DSC
- **Built-in knowledgebase**, which immediately flags resource conflicts and incorrect settings

Creating…

- **Hardware Abstraction Layer (HAL)** – hardware-dependent, low-level drivers with a known application programming interface (API)

Benefits

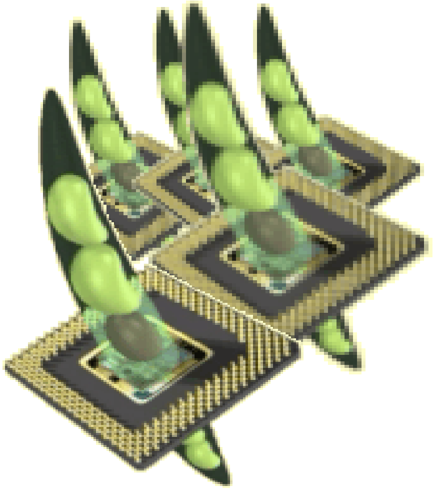
- Eases migration between Freescale devices
- Designers don't have to be intimately familiar with every page of a specification
- Errors are caught early in design cycle; therefore, designers get to market faster with higher quality product

CodeWarrior / Processor Expert Support

- ►Processor Expert is integrated into the CodeWarrior tool suite with support for
- ►CodeWarrior Development Studio for 56800e Digital Signal Controllers (DSC)
  - 56800: 5680x, 5682x, 5685x
  - 56800e: 56801x, 56802x, 56803x, 5681xx, 5683xx

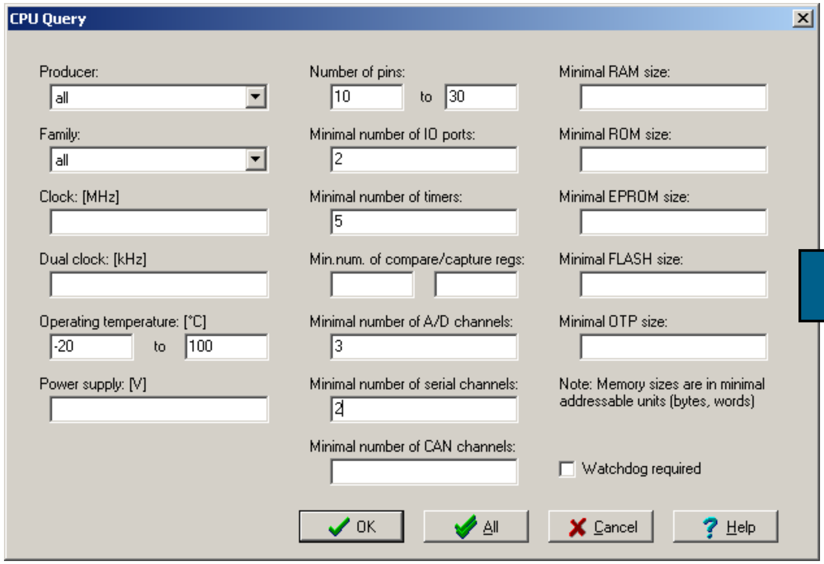
What is an Embedded Bean?

- Embedded Beans are software components, which encapsulate the initialization and functionality of an embedded system’s basic elements
  - CPU core
  - CPU on-chip peripherals
  - Stand-alone peripherals
  - Virtual devices
  - Pure software algorithms
- Embedded Beans provide a hardware abstraction layer (HAL), which eases migration between devices



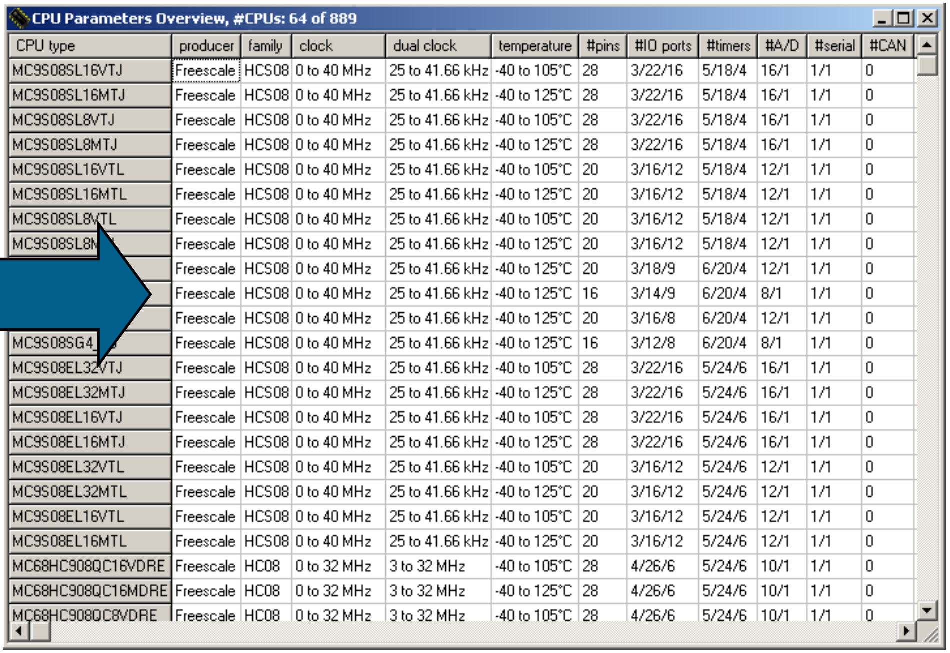
Silicon Selection

- You can access the knowledgebase in Processor Expert to find Freescale silicon that will meet your application needs
  - Select CPU Parameters Overview in the Processor Expert > View menu.



CPU Query

Producer: all	Number of pins: 10 to 30	Minimal RAM size:
Family: all	Minimal number of IO ports: 2	Minimal ROM size:
Clock: [MHz]	Minimal number of timers: 5	Minimal EPROM size:
Dual clock: [kHz]	Min.num. of compare/capture regs:	Minimal FLASH size:
Operating temperature: [°C] -20 to 100	Minimal number of A/D channels: 3	Minimal OTP size:
Power supply: [V]	Minimal number of serial channels: 2	Note: Memory sizes are in minimal addressable units (bytes, words)
	Minimal number of CAN channels:	□ Watchdog required

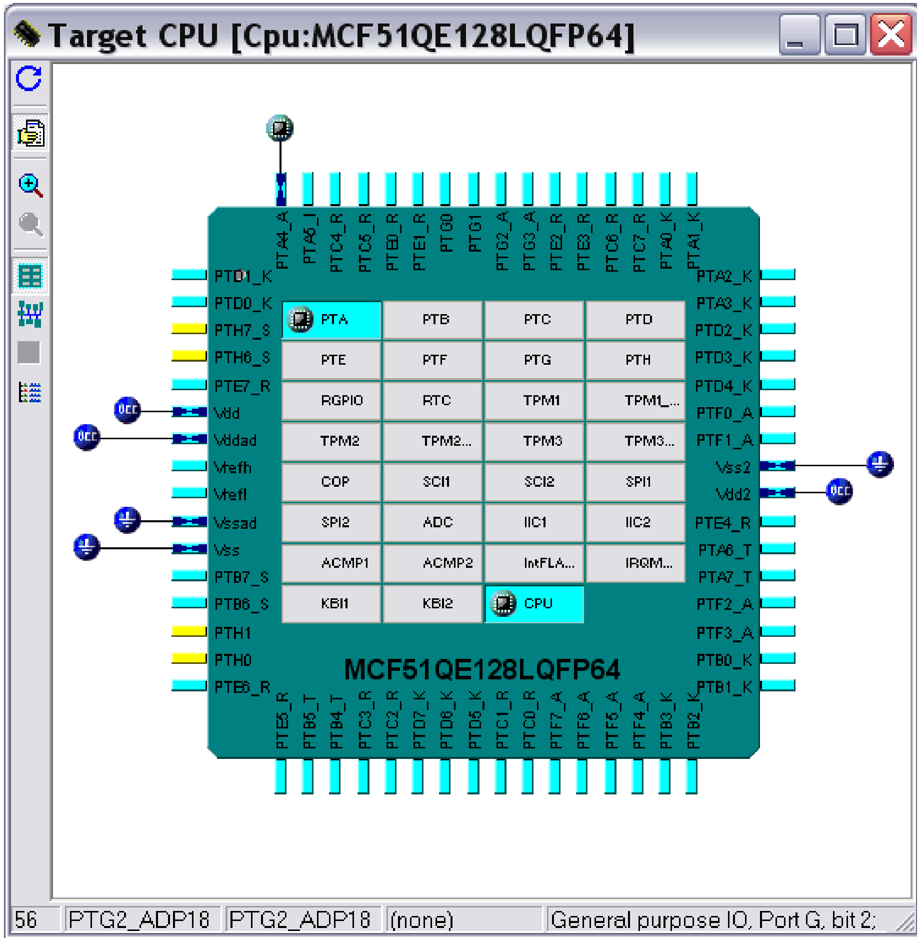


CPU Parameters Overview, #CPUs: 64 of 889

CPU type	producer	family	clock	dual clock	temperature	#pins	#IO ports	#timers	#A/D	#serial	#CAN
MC9S08SL16VTJ	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 105°C	28	3/22/16	5/18/4	16/1	1/1	0
MC9S08SL16MTJ	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 125°C	28	3/22/16	5/18/4	16/1	1/1	0
MC9S08SL8VTJ	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 105°C	28	3/22/16	5/18/4	16/1	1/1	0
MC9S08SL8MTJ	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 125°C	28	3/22/16	5/18/4	16/1	1/1	0
MC9S08SL16VTL	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 105°C	20	3/16/12	5/18/4	12/1	1/1	0
MC9S08SL16MTL	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 125°C	20	3/16/12	5/18/4	12/1	1/1	0
MC9S08SL8VTL	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 105°C	20	3/16/12	5/18/4	12/1	1/1	0
MC9S08SL8M[illegible]	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 125°C	20	3/16/12	5/18/4	12/1	1/1	0
[illegible]	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 125°C	20	3/18/9	6/20/4	12/1	1/1	0
[illegible]	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 125°C	16	3/14/9	6/20/4	8/1	1/1	0
[illegible]	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 125°C	20	3/16/8	6/20/4	12/1	1/1	0
MC9S08SG4_[illegible]	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 125°C	16	3/12/8	6/20/4	8/1	1/1	0
MC9S08EL32VTJ	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 105°C	28	3/22/16	5/24/6	16/1	1/1	0
MC9S08EL32MTJ	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 125°C	28	3/22/16	5/24/6	16/1	1/1	0
MC9S08EL16VTJ	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 105°C	28	3/22/16	5/24/6	16/1	1/1	0
MC9S08EL16MTJ	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 125°C	28	3/22/16	5/24/6	16/1	1/1	0
MC9S08EL32VTL	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 105°C	20	3/16/12	5/24/6	12/1	1/1	0
MC9S08EL32MTL	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 125°C	20	3/16/12	5/24/6	12/1	1/1	0
MC9S08EL16VTL	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 105°C	20	3/16/12	5/24/6	12/1	1/1	0
MC9S08EL16MTL	Freescale	HCS08	0 to 40 MHz	25 to 41.66 kHz	-40 to 125°C	20	3/16/12	5/24/6	12/1	1/1	0
MC68HC908QC16VDRE	Freescale	HC08	0 to 32 MHz	3 to 32 MHz	-40 to 105°C	28	4/26/6	5/24/6	10/1	1/1	0
MC68HC908QC16MDRE	Freescale	HC08	0 to 32 MHz	3 to 32 MHz	-40 to 125°C	28	4/26/6	5/24/6	10/1	1/1	0
MC68HC908QC8VDRE	Freescale	HC08	0 to 32 MHz	3 to 32 MHz	-40 to 105°C	28	4/26/6	5/24/6	10/1	1/1	0

Silicon Evaluation

- You can use Target CPU Window to evaluate silicon
  - Displays selected target microcontroller with its peripherals and pins
  - Displays current resource usage by selected beans (i.e. peripherals, pins)
  - Data directions of single pins are indicated by blue arrows when configured by a bean
  - Pins associated with a peripheral are highlighted when mouse hovers over a peripheral
  - Help is available for pins and peripherals by moving the mouse over the item



Target CPU [Cpu:MCF51QE128LQFP64]

PTA	PTB	PTC	PTD
PTE	PTF	PTG	PTH
RGPIO	RTC	TPM1	TPM1_...
TPM2	TPM2...	TPM3	TPM3...
COP	SCI1	SCI2	SPI1
SPI2	ADC	IIC1	IIC2
ACMP1	ACMP2	IntFLA...	IRQM...
KBI1	KBI2	CPU	

56 | PTG2_ADP18 | PTG2_ADP18 | (none) | General purpose IO, Port G, bit 2;

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © Freescale Semiconductor, Inc. 2009.

Agenda

- FreeMASTER Overview
- Quick_Start Overview
- Processor Exert Overview
- Processor Expert Demo

Processor Expert LIVE DEMO

- TIMER/LED CODE GENERATED FROM SCRATCH
  - Stationery selected from Processor Expert repertoire
  - Beans added for LED and TIMER
  - LED Method dragged-and-dropped into TIMER EVENT
  - Code built and run

Q&A

- Thank you for attending this presentation. We’ll now take a few moments for the audience’s questions and then we’ll begin the question and answer session.

freescale™
semiconductor