EVROPSKÁ UNIE

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

OP Vzdělávání pro konkurenceschopnost

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Mikroprocesorová technika v embedded systémech

Ing. Jaroslav Lepka
Ing. Libor Prokop

27. – 28. května 2010

Tato prezentace je spolufinancována Evropským sociálním fondem a státním rozpočtem České republiky.

CodeWarrior™ Integrated Development Environment (IDE)

CodeWarrior IDE

- Efficient and flexible software-development tool suite
- Consists of
  - Project manager
  - Graphical user interface
  - Compilers
  - Linkers
  - Debuggers
  - Source code browser
  - Editing tools

CodeWarrior IDE – DSP56800E

- **CodeWarrior Compiler for DSP56800E**
  - ANSI-compliant C compiler
  - Based on the same compiler architecture used in all CodeWarrior C compilers
  - Use this compiler with the CodeWarrior linker for DSP56800E to generate DSP56800E applications and libraries
- **CodeWarrior Linker for DSP56800E**
  - Lets us generate
    - Either Executable and Linker Format (ELF)
    - Or S-record output files for your application

CodeWarrior IDE – DSP56800E

- **CodeWarrior Assembler for DSP56800E**
  - Easy-to-use syntax
  - It assembles any project file that has a.asm filename
- **Main Standard Library (MSL)**
  - Set of ANSI-compliant, standard C libraries for use in developing DSP56800E applications
  - Subset of those used for all platform targets
    - These libraries are customized and the runtime adapted for DSP56800E development

CodeWarrior IDE – DSP56800E

- **CodeWarrior Debugger for DSP56800E**
  - Controls your program’s execution, letting you see what happens internally as your program runs
  - Debugger can
    - Execute your program one statement at a time, suspending execution when control reaches a specified point
    - Show the chain of function calls
    - Examine and change the values of variables
    - Inspect processor register contents
    - Watch the contents of memory
    - Etc.
  - Use this debugger to find problems in your program

CodeWarrior IDE

- **Target settings**
  - Each build target in a CodeWarrior™ project has its own settings
  - The target settings control:
    - Compiler options
    - Linker options
    - Assembler options
    - Debugger options
    - Error and warning messages
  - When you create a project using stationery
    - The build targets, which are part of the stationery, already include default target settings
    - You can use those default target settings (if the settings are appropriate), or you can change them

CodeWarrior IDE – Debugger

- Manipulates Program Execution
  - Breakpoints
    - Halt program execution on a line of source code that you specify
    - Two types of brakepoint
      - Always halts program execution
      - Halts program execution if a condition that you specify is true
  - Watchpoints
    - Halt program execution after a location in memory is accessed

CodeWarrior IDE – Debugger

- Manipulates Program Execution
  - Eventpoints
    - Perform a task during program execution on a line of source code that you specify
    - Eventpoints can play sounds, run scripts, log data, and perform other operations
  - Special breakpoints
    - These internal breakpoints halt program execution in special cases
      - such as halting program execution at the main() function or for a C++ exception. Halt program execution after a location in memory is accessed

Quick Start Tool

Agenda

1. What is Quick_Start?
2. Quick_Start Low-level Drivers
3. Graphical Configuration Tool
4. Demo
5. Discussion

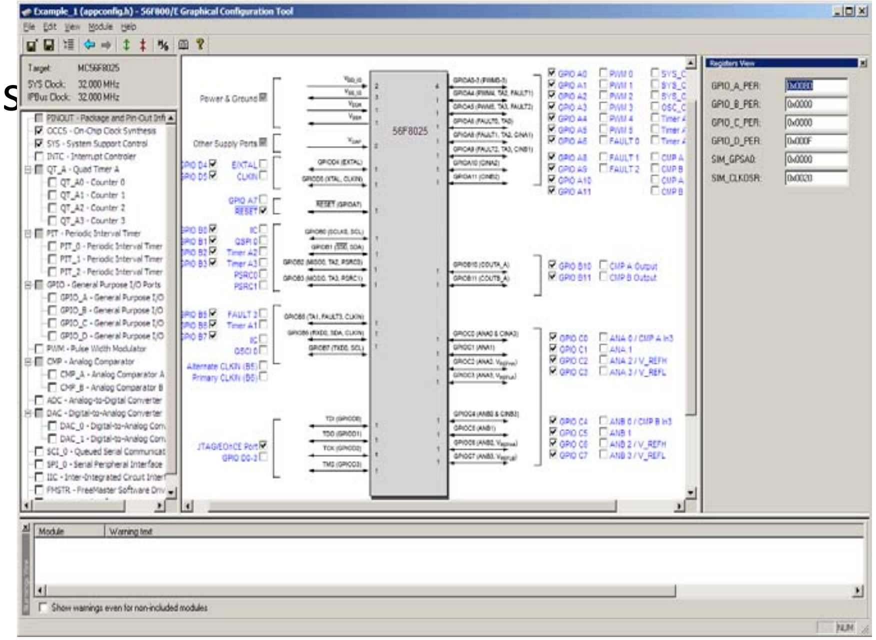
What is Quick Start?

Quick Start = Easy-to-use SW 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
- **Sample Applications**
  - Demonstrating usage of GCT, processor peripheral modules and low-level drivers
- **User Manual**
  - Low-level drivers & tools guide
  - Latest device User Manual



Example_1 (appconfig.h) - 56F800/E Graphical Configuration Tool

Target: MC56F8025
SYS Clock: 32.000 MHz
IPBus Clock: 32.000 MHz

Registers View

GPIO_A_PER: [illegible]
GPIO_B_PER: 0x0000
GPIO_C_PER: 0x0000
GPIO_D_PER: 0x000F
SIM_GPSA0: 0x0000
SIM_CLKOSR: 0x0020

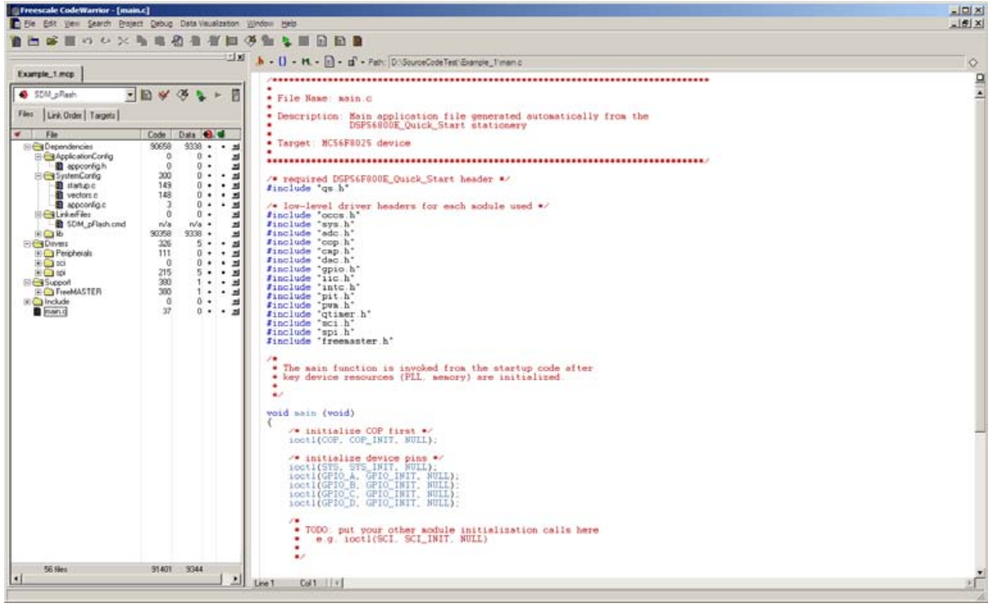
□ Show warnings even for non-included modules

File Edit View Module Help
□ PINOUT - Package and Pin-Out Info
☑ OCCS - On-Chip Clock Synthesis
☑ SYS - System Support Control
□ INTC - Interrupt Controller
QT_A - Quad Timer A
PIT - Periodic Interval Timer
GPIO - General Purpose I/O Ports
PWM - Pulse Width Modulator
CMP - Analog Comparator
ADC - Analog-to-Digital Converter
DAC - Digital-to-Analog Converter
SCI_0 - Queued Serial Communications
SPI_0 - Serial Peripheral Interface
IIC - Inter-Integrated Circuit Interface
FMSTR - FreeMaster Software Driver
Power & Ground
Other Supply Ports
56F8025
JTAG/EOnCE Port
Module
Warning text
NUM

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


Freescale CodeWarrior - [main.c]
File Edit View Search Project Debug Data Visualization Window Help
Example_1.mcp
SDM_pFlash
Files Link Order Targets
Path: D:\SourceCodeTest\Example_1\main.c
Line 1 Col 1

File	Code	Data
Dependencies	90658	9338
ApplicationConfig	0	0
appconfig.h	0	0
SystemConfig	300	0
startup.c	149	0
vectors.c	148	0
appconfig.c	3	0
LinkerFiles	0	0
SDM_pFlash.cmd	n/a	n/a
lib	90358	9338
Drivers	326	5
Peripherals	111	0
sci	0	0
spi	215	5
Support	380	1
FreeMASTER	380	1
Include	0	0
main.c	37	0
56 files	91401	9344

```
/**********************************************************************
*
* File Name: main.c
*
* Description: Main application file generated automatically from the
*              DSP56800E_Quick_Start stationery
*
* Target: MC56F8025 device
*
**********************************************************************/

/* required DSP56F800E_Quick_Start header */
#include "qs.h"

/* low-level driver headers for each module used */
#include "occs.h"
#include "sys.h"
#include "adc.h"
#include "cop.h"
#include "cmp.h"
#include "dac.h"
#include "gpio.h"
#include "iic.h"
#include "intc.h"
#include "pit.h"
#include "pwm.h"
#include "qtimer.h"
#include "sci.h"
#include "spi.h"
#include "freemaster.h"

/*
 * The main function is invoked from the startup code after
 * key device resources (PLL, memory) are initialized.
 *
 */

void main (void)
{
    /* initialize COP first */
    ioctl(COP, COP_INIT, NULL);

    /* initialize device pins */
    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);

    /*
     * TODO: put your other module initialization calls here
     *   e.g. ioctl(SCI, SCI_INIT, NULL)
     *
     */
```

ArchIO Structure

- *ArchIO* – global symbol
  - Provides a C interface (structure type) to all peripheral and core registers mapped in data memory
  - All registers are accessed via this structure - no need to know and specify the concrete addresses of the registers to write or read
  - *ArchIO* - declared in the *arch.h* file
  - *ArchIO* structure definition
    - *ArchIO* defined as the *extern* variable
    - Its address defined by a directive in linker command file

ArchIO Structure

```
typedef volatile struct
{
    arch_sTimer    TimerA;         /* TMRA_BASE    0xF000 */
    arch_sTimer    TimerB_unused;
    arch_sADC      Adc;            /* ADC_BASE     0xF080 */
    arch_sPWM      Pwm;            /* PWM_BASE     0xF0C0 */
    arch_sIntc     Intc;           /* INTC_BASE    0xF0E0 */
    arch_sSIM      Sim;            /* SIM_BASE     0xF100 */
    arch_sCOP      Cop;            /* COP_BASE     0xF120 */
    arch_sPLL      Pll;            /* PLL_BASE     0xF130 */
    arch_sLVI      Lvi;            /* LVI_BASE     0xF140 */
    .
    .
    UWord16        reserved4[0xFF0600];
    arch_sEOnCE    EOnCE;          /* EOnCE_BASE   0xFFFF00 */
} arch_sIO;
```

ArchIO Structure

- COP structure – defined in arch.h file

```
typedef volatile struct
{
    ARCH_REG2(UWord16, copctl,   ControlReg);
    ARCH_REG2(UWord16, copto,    TimeoutReg);
    ARCH_REG2(UWord16, copctr,   ServiceReg);
    ARCH_REG1(UWord16, reserved[13]);
} arch_sCOP;
```

ArchIO Structure

- arch.h file – extern declaration of ArchIO variable

```
/* The location of the following structure is defined in linker.cmd
   */
  extern arch_sIO    ArchIO;
```

- Linker command file – address assignment to the structure

```
FArchIO = ADDR(.x_onchip_peripherals);
```

Using the ArchIO Structure

- Example of read/write operation using ArchIO structure

```
UWord16 RegValue;    // variable definition
RegValue = ArchIO.TimerA.Channel0.HoldReg;  // read register
ArchIO.TimerA.Channel0.CompareReg1 = 0x8000;  // write number to reg
```

- Example of the same operation as previous case using *periphMemRead* and *periphMemRead* macros

```
UWord16 RegValue;    // variable definition
RegValue = periphMemRead(&ArchIO.TimerA.Channel0.HoldReg)
periphMemWrite(0x8000,  &ArchIO.TimerA.Channel0.CompareReg1 = 0x8000);
```

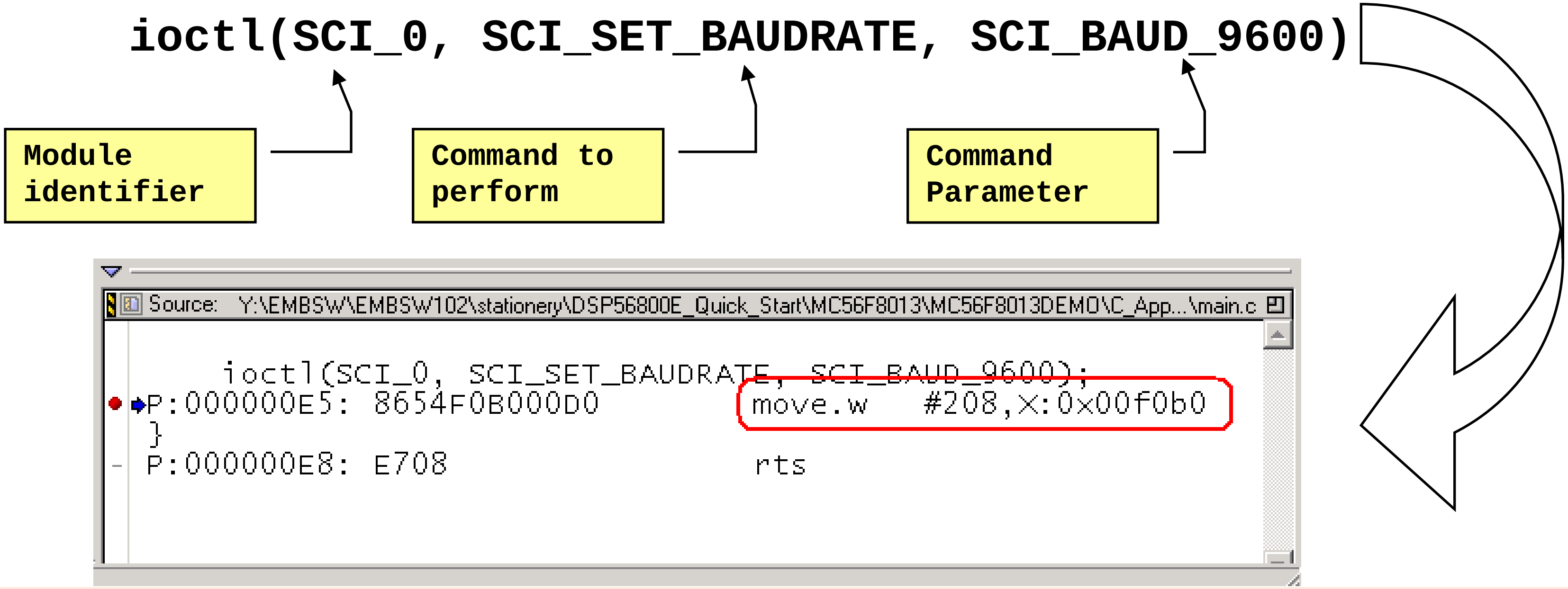
Low-Level Drivers

56F800/E, MPC500, MPC5500

QUICK START OVERVIEW

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
```

ioctl Commands

- ioctl – Input Output Control
- ioctl – general syntax

  ioctl(module_ID, cmd_name, cmd_spec_param);
- **module_ID** – module identifier
  - Predefined symbolic constant corresponding to names of peripheral modules
    - Example: GPIO_A, GPIO_B, ADC, ADC_A, ADC_B, PWM, PWM_A, PWM_B, COP, etc.
  - The base address of the peripheral module
  - List of module identifiers – “*.h” corresponding to managed peripheral
    - Example: gpio.h, adc.h, pwm.h, sci.h, spi.h, qtimer.h, etc.

ioctl Commands

- **cmd_name** – specifies action performed on a peripheral module
  - Command is depended to performed operation
  - List of commands – “*.h” corresponding to managed peripheral
    - Example: gpio.h, adc.h, pwm.h, sci.h, spi.h, qtimer.h, etc.
  - Set of commands for each peripheral
    - Example for pwm.h:
      - PWM_SET_PRESCALER
      - PWM_SET_RELOAD_FREQUENCY
      - PWM_FAULT_INT_ENABLE
      - Etc.
    - Self-explaining name of commands
    - No need to dive into deep documentation studying
  - INIT command – essential command for each peripheral
    - Example: COP_INIT, ADC_INIT, PWM_INIT, GPIO_INIT, etc.

ioctl Commands

- cmd_spec_param – command specific parameter
  - Specifies other data required to execute the command
  - In general, it can be
    - Pointer to the structure
    - NULL value
    - Variable-value in dependency with the specific command
  - List of recommended parameters – “*.h” corresponding to managed peripheral
    - Example: gpio.h, adc.h, pwm.h, sci.h, spi.h, qtimer.h, etc.
    - Example for pwm.h:
      - #define PWM_PRESCALER_DIV_1   0
      - #define PWM_PRESCALER_DIV_2   1
      - #define PWM_PRESCALER_DIV_4   2
      - #define PWM_PRESCALER_DIV_8   3
      - Etc.

ioctl Commands Implementation

- ioctl command - macro

```
#define ioctl(fd,cmd,prm) ioctl##cmd((fd),(prm))
```

- Macro definition – periph.h
- fd
  - Peripheral module base address
  - Address assigned from ArchIO structure

ioctl Commands Implementation

- Example for GPIO – general command
  - gpio.h
    - **#define GPIO_A (&ArchIO.PortA) // GPIO_A base address**
  - User source code - *.c
    - **ioctl(GPIO_A, GPIO_SET_PIN, BIT_0);**
  - periph.h
    - **#define periphBitSet(mask, addr) (*(addr) |= (mask))**
  - gpio.h
    - **#define ioctlGPIO_SET_PIN(pGpioBase, param)**

      periphBitSet(param, &((pGpioBase)->dr))
  - Compiler result – assembly code

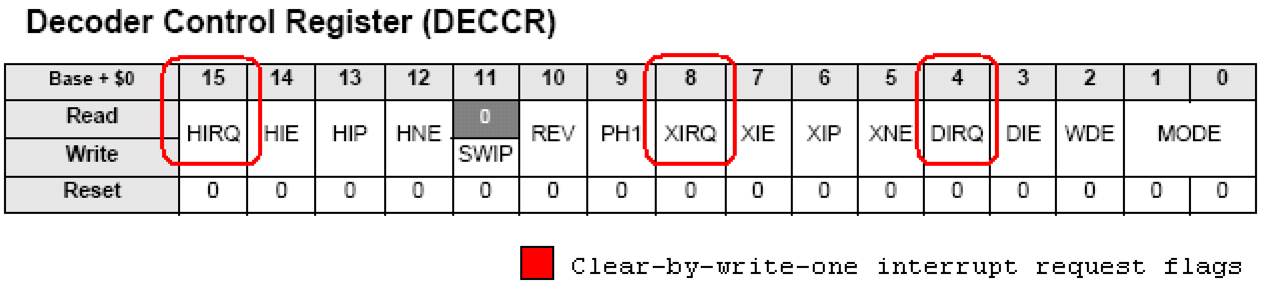
    ```
    ioctl(GPIO_A, GPIO_SET_PIN, BIT_0);
    P:0000414A: 8254F1510001      bfset   #1,X:0x00f151
    ```

ioctl Commands Implementation

- Example for GPIO – INIT command
  - gpio.h
    - **#define GPIO_A (&ArchIO.PortA) // GPIO_A base address**
  - User source code - *.c
    - **ioctl(GPIO_A, GPIO_INIT, NULL);**
  - gpio.h
    - **void gpioInit(arch_sPort *pGpioBase); // declaration**
    - **#define ioctlGPIO_INIT(pGpioBase, param) gpioInit(pGpioBase)**
  - gpioInit() function execution
    - Function definition - gpio.c
    - Usually executed just ones during chip initialization
    - Performs setting stored in appconfig.h file
    - appconfig.h file modified by GCT (Graphical Configuration Tool)

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)

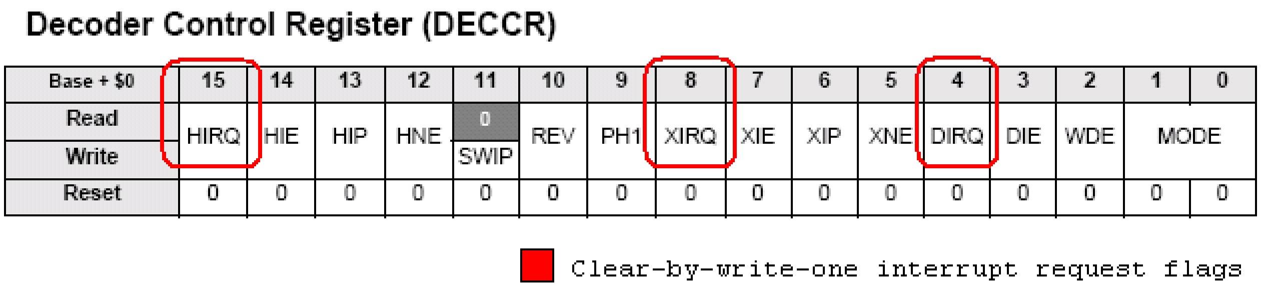
Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HIRQ	HIE	HIP	HNE	0	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	MODE	
Write					SWIP											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

🟥 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...

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;
```

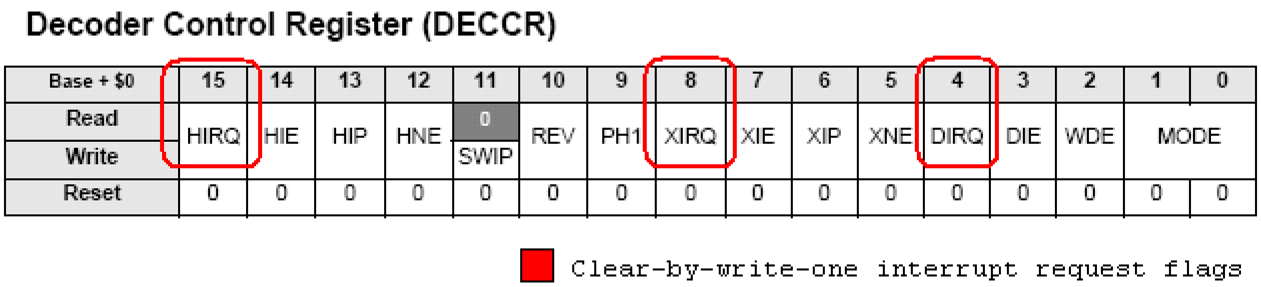
?

```
56F800E Assembler:

    asm ( 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	HIRQ	HIE	HIP	HNE	SWIP	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	MODE	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

🟥 Clear-by-write-one interrupt request flags

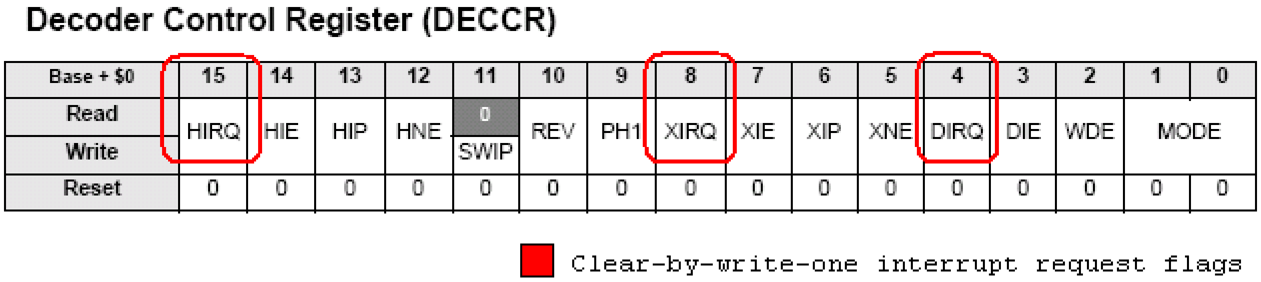
```
#define DECCR_DIRQ 0x0010   /* DIRQ bit constant */
ArchIO.Decoder0.deccr       /* register in the peripheral structure */
```

```
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

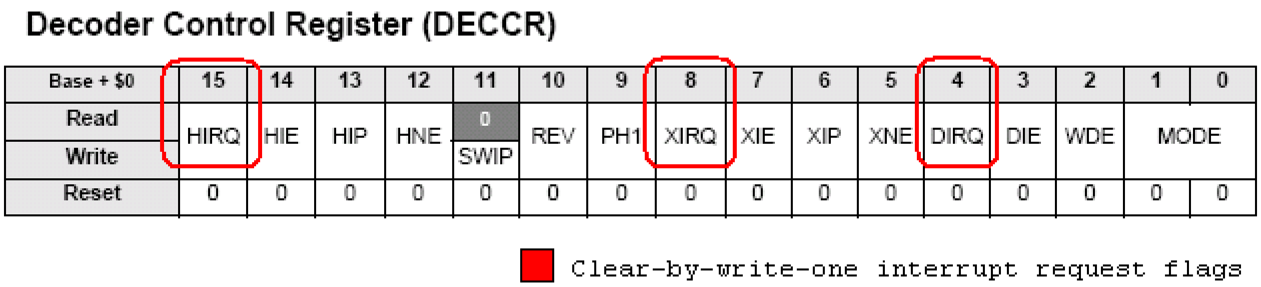
🟥 Clear-by-write-one interrupt request flags

```
#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) &
```

56F8

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

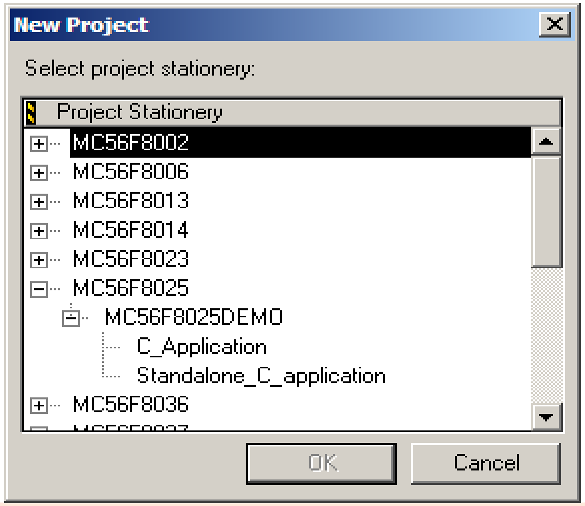
56F800/E, MPC500, MPC5500, MPC5200

QUICK START OVERVIEW

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
- MC56F8002
- MC56F8006
- MC56F8013
- MC56F8014
- MC56F8023
- MC56F8025
  - MC56F8025DEMO
    - C_Application
    - Standalone_C_application
- MC56F8036

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

Graphical Configuration Tool

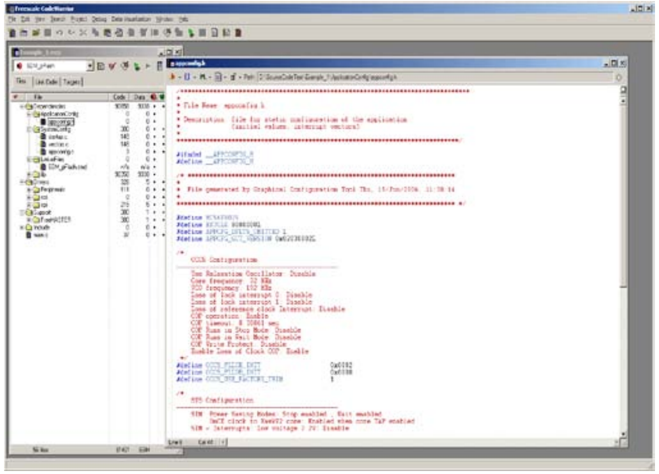
56F800/E, MPC500, MPC5500, MPC5200

QUICK START OVERVIEW

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)

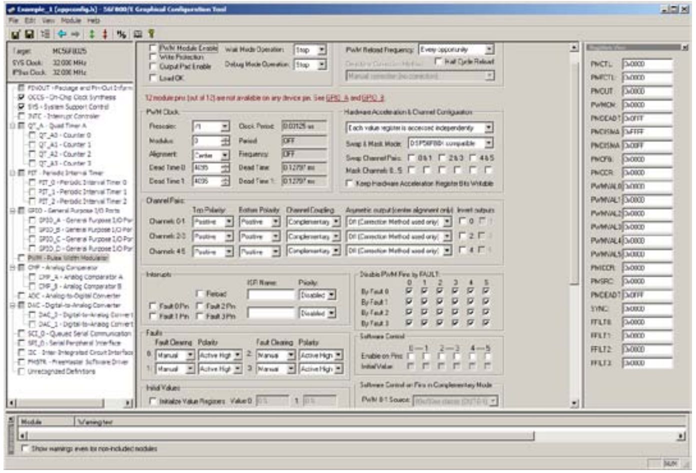


Metrowerks CodeWarrior IDE

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

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

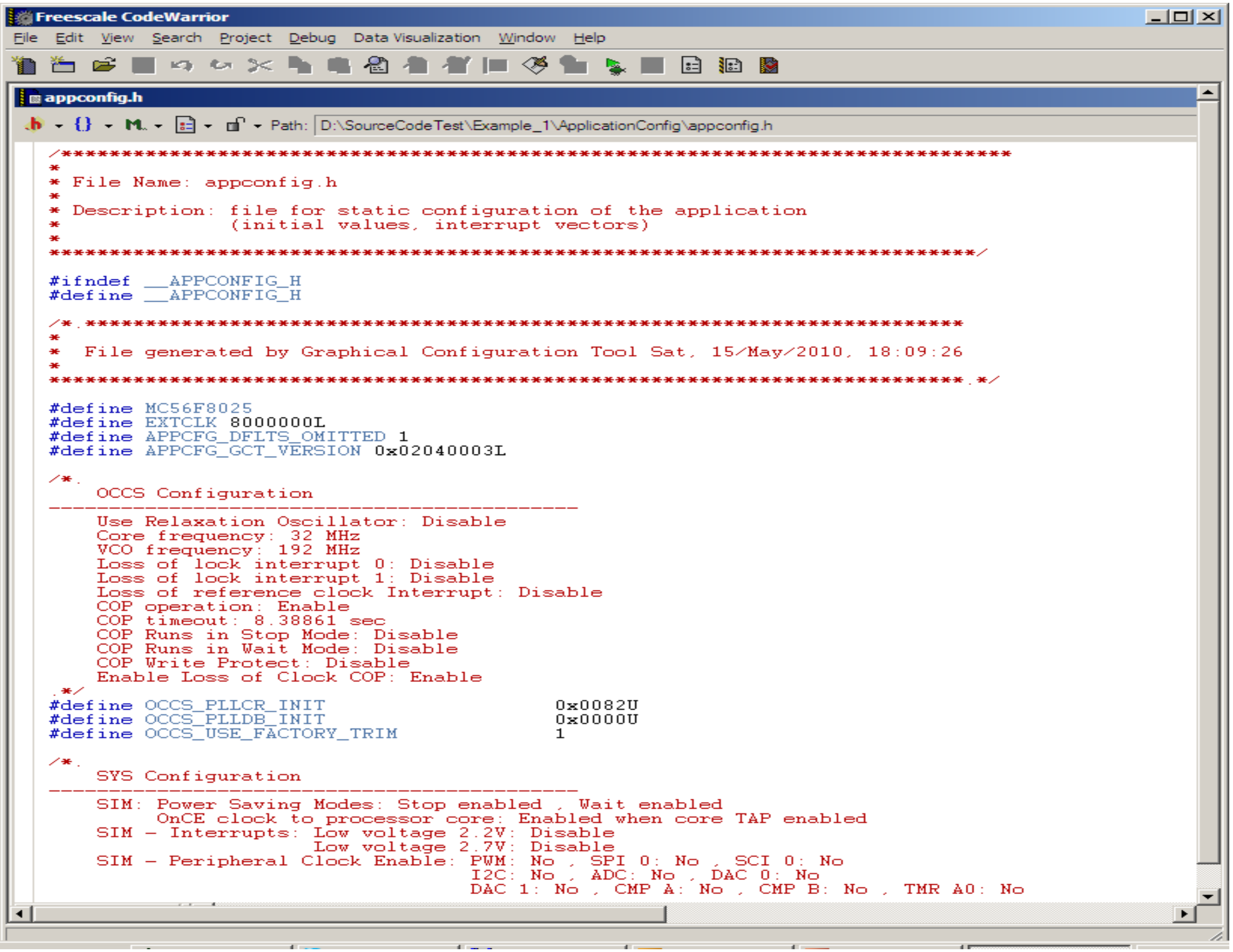
appconfig.h file



Graphical Configuration Tool

Read & Write access to appconfig.h

Graphical Configuration Tool: appconfig.h


Freescale CodeWarrior
File Edit View Search Project Debug Data Visualization Window Help
appconfig.h
Path: D:\SourceCodeTest\Example_1\ApplicationConfig\appconfig.h

```
/*****************************************************************************
*
* File Name: appconfig.h
*
* Description: file for static configuration of the application
*              (initial values, interrupt vectors)
*
****************************************************************************/

#ifndef __APPCONFIG_H
#define __APPCONFIG_H

/*.*************************************************************************
*
*  File generated by Graphical Configuration Tool Sat, 15/May/2010, 18:09:26
*
*************************************************************************.*/

#define MC56F8025
#define EXTCLK 8000000L
#define APPCFG_DFLTS_OMITTED 1
#define APPCFG_GCT_VERSION 0x02040003L

/*.
    OCCS Configuration
--------------------------------------------
    Use Relaxation Oscillator: Disable
    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: Enable
    COP timeout: 8.38861 sec
    COP Runs in Stop Mode: Disable
    COP Runs in Wait Mode: Disable
    COP Write Protect: Disable
    Enable Loss of Clock COP: Enable
.*/
#define OCCS_PLLCR_INIT                        0x0082U
#define OCCS_PLLDB_INIT                        0x0000U
#define OCCS_USE_FACTORY_TRIM                  1

/*.
    SYS Configuration
--------------------------------------------
    SIM: Power Saving Modes: Stop enabled , Wait enabled
         OnCE clock to processor core: Enabled when core TAP enabled
    SIM - Interrupts: Low voltage 2.2V: Disable
                      Low voltage 2.7V: Disable
    SIM - Peripheral Clock Enable: PWM: No , SPI 0: No , SCI 0: No
                                   I2C: No , ADC: No , DAC 0: No
                                   DAC 1: No , CMP A: No , CMP B: No , TMR A0: No
```

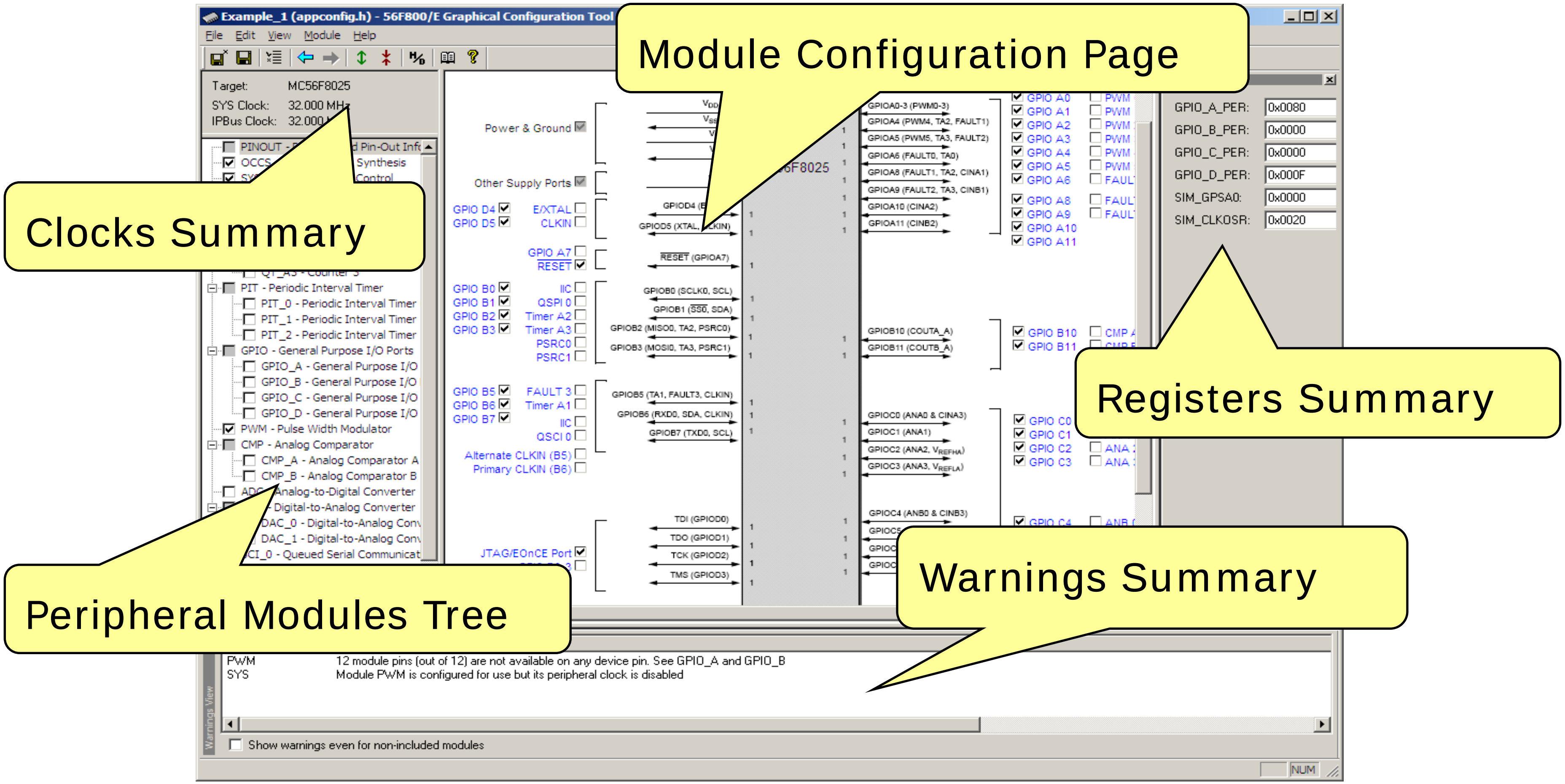
Graphical Configuration Tool: appconfig.h

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**



Example_1 (appconfig.h) - 56F800/E Graphical Configuration Tool

File Edit View Module Help

Target: MC56F8025
SYS Clock: 32.000 MHz
IPBus Clock: 32.000 M

- □ PINOUT - P... d Pin-Out Infc
- ☑ OCCS ... Synthesis
- ☑ SY... Control
- □ QT_A3 - Counter 3
- □ 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
- □ ... - Digital-to-Analog Converter
  - □ DAC_0 - Digital-to-Analog Conv
  - □ DAC_1 - Digital-to-Analog Conv
- ...CI_0 - Queued Serial Communicat

GPIO_A_PER:	0x0080
GPIO_B_PER:	0x0000
GPIO_C_PER:	0x0000
GPIO_D_PER:	0x000F
SIM_GPSA0:	0x0000
SIM_CLKOSR:	0x0020

PWM 12 module pins (out of 12) are not available on any device pin. See GPIO_A and GPIO_B
SYS Module PWM is configured for use but its peripheral clock is disabled

□ Show warnings even for non-included modules

Module Configuration Page

Clocks Summary

Registers Summary

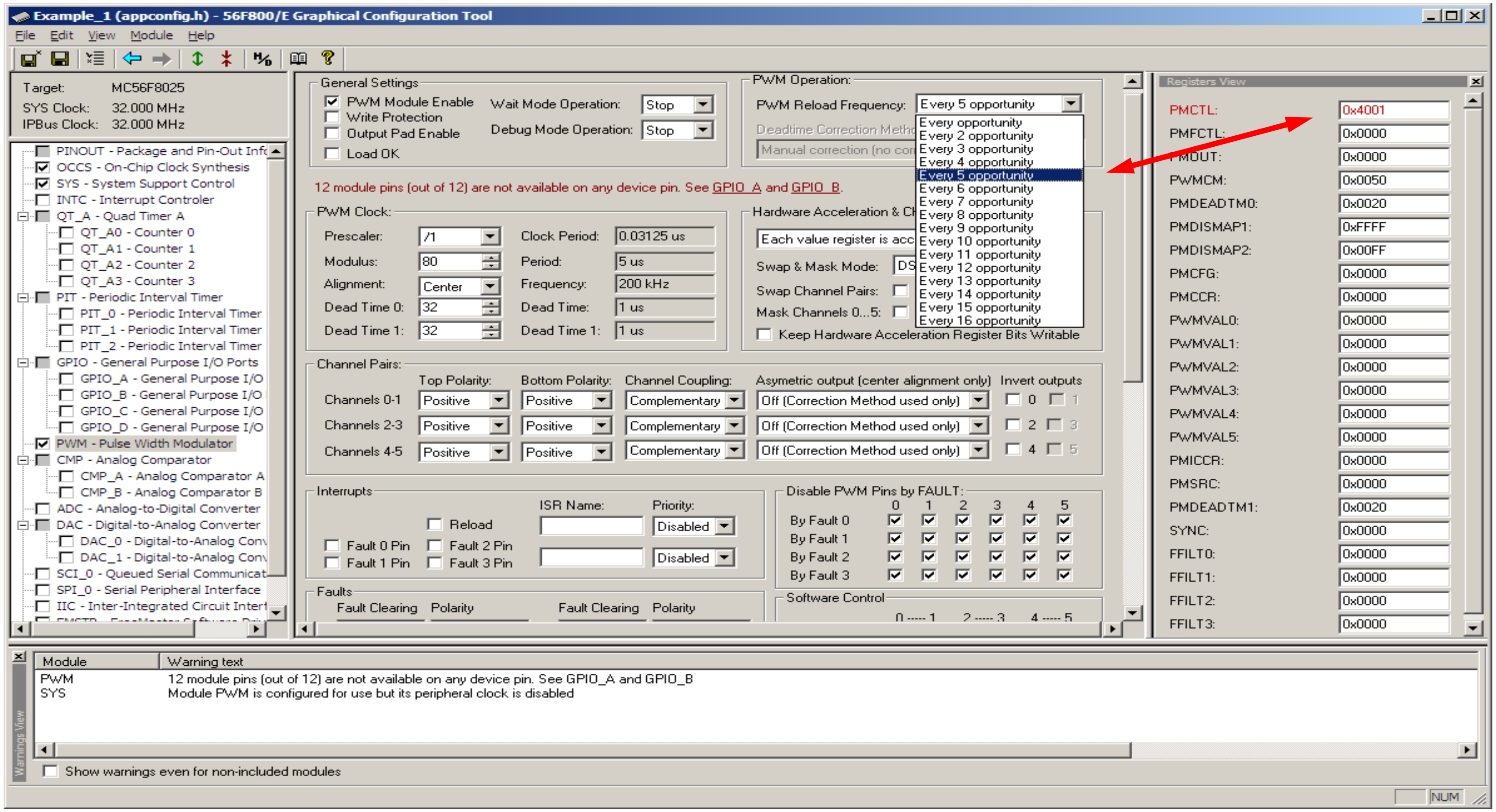
Warnings Summary

Peripheral Modules Tree

28.5.2010

Graphical Configuration Tool

- **Direct Register Value View**



Example_1 (appconfig.h) - 56F800/E Graphical Configuration Tool

Target: MC56F8025
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_A - Quad Timer A
  - □ QT_A0 - Counter 0
  - □ QT_A1 - Counter 1
  - □ QT_A2 - Counter 2
  - □ QT_A3 - Counter 3
- □ 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
  - □ DAC_1 - Digital-to-Analog Conv
- □ SCI_0 - Queued Serial Communicat
- □ SPI_0 - Serial Peripheral Interface
- □ IIC - Inter-Integrated Circuit Interf

General Settings
- ☑ PWM Module Enable
- □ Write Protection
- □ Output Pad Enable
- □ Load OK

Wait Mode Operation: Stop
Debug Mode Operation: Stop

PWM Operation:
PWM Reload Frequency: Every 5 opportunity
Deadtime Correction Meth
Manual correction (no cor

Every opportunity
Every 2 opportunity
Every 3 opportunity
Every 4 opportunity
Every 5 opportunity
Every 6 opportunity
Every 7 opportunity
Every 8 opportunity
Every 9 opportunity
Every 10 opportunity
Every 11 opportunity
Every 12 opportunity
Every 13 opportunity
Every 14 opportunity
Every 15 opportunity
Every 16 opportunity

12 module pins (out of 12) are not available on any device pin. See GPIO_A and GPIO_B.

PWM Clock:

Prescaler:	/1	Clock Period:	0.03125 us
Modulus:	80	Period:	5 us
Alignment:	Center	Frequency:	200 kHz
Dead Time 0:	32	Dead Time:	1 us
Dead Time 1:	32	Dead Time 1:	1 us

Hardware Acceleration & Ch
Each value register is acc
Swap & Mask Mode: DS
Swap Channel Pairs: □
Mask Channels 0...5: □
□ Keep Hardware Acceleration Register Bits Writable

Channel Pairs:

	Top Polarity:	Bottom Polarity:	Channel Coupling:	Asymetric output (center alignment only)	Invert outputs
Channels 0-1	Positive	Positive	Complementary	Off (Correction Method used only)	□ 0 □ 1
Channels 2-3	Positive	Positive	Complementary	Off (Correction Method used only)	□ 2 □ 3
Channels 4-5	Positive	Positive	Complementary	Off (Correction Method used only)	□ 4 □ 5

Interrupts
□ Reload
□ Fault 0 Pin □ Fault 2 Pin
□ Fault 1 Pin □ Fault 3 Pin
ISR Name: Priority: Disabled
Priority: Disabled

Disable PWM Pins by FAULT:

	0	1	2	3	4	5
By Fault 0	☑	☑	☑	☑	☑	☑
By Fault 1	☑	☑	☑	☑	☑	☑
By Fault 2	☑	☑	☑	☑	☑	☑
By Fault 3	☑	☑	☑	☑	☑	☑

Faults
Fault Clearing Polarity Fault Clearing Polarity

Software Control
0 ----- 1 2 ----- 3 4 ----- 5

Registers View

PMCTL:	0x4001
PMFCTL:	0x0000
PMOUT:	0x0000
PWMCM:	0x0050
PMDEADTM0:	0x0020
PMDISMAP1:	0xFFFF
PMDISMAP2:	0x00FF
PMCFG:	0x0000
PMCCR:	0x0000
PWMVAL0:	0x0000
PWMVAL1:	0x0000
PWMVAL2:	0x0000
PWMVAL3:	0x0000
PWMVAL4:	0x0000
PWMVAL5:	0x0000
PMICCR:	0x0000
PMSRC:	0x0000
PMDEADTM1:	0x0020
SYNC:	0x0000
FFILT0:	0x0000
FFILT1:	0x0000
FFILT2:	0x0000
FFILT3:	0x0000

Warnings View

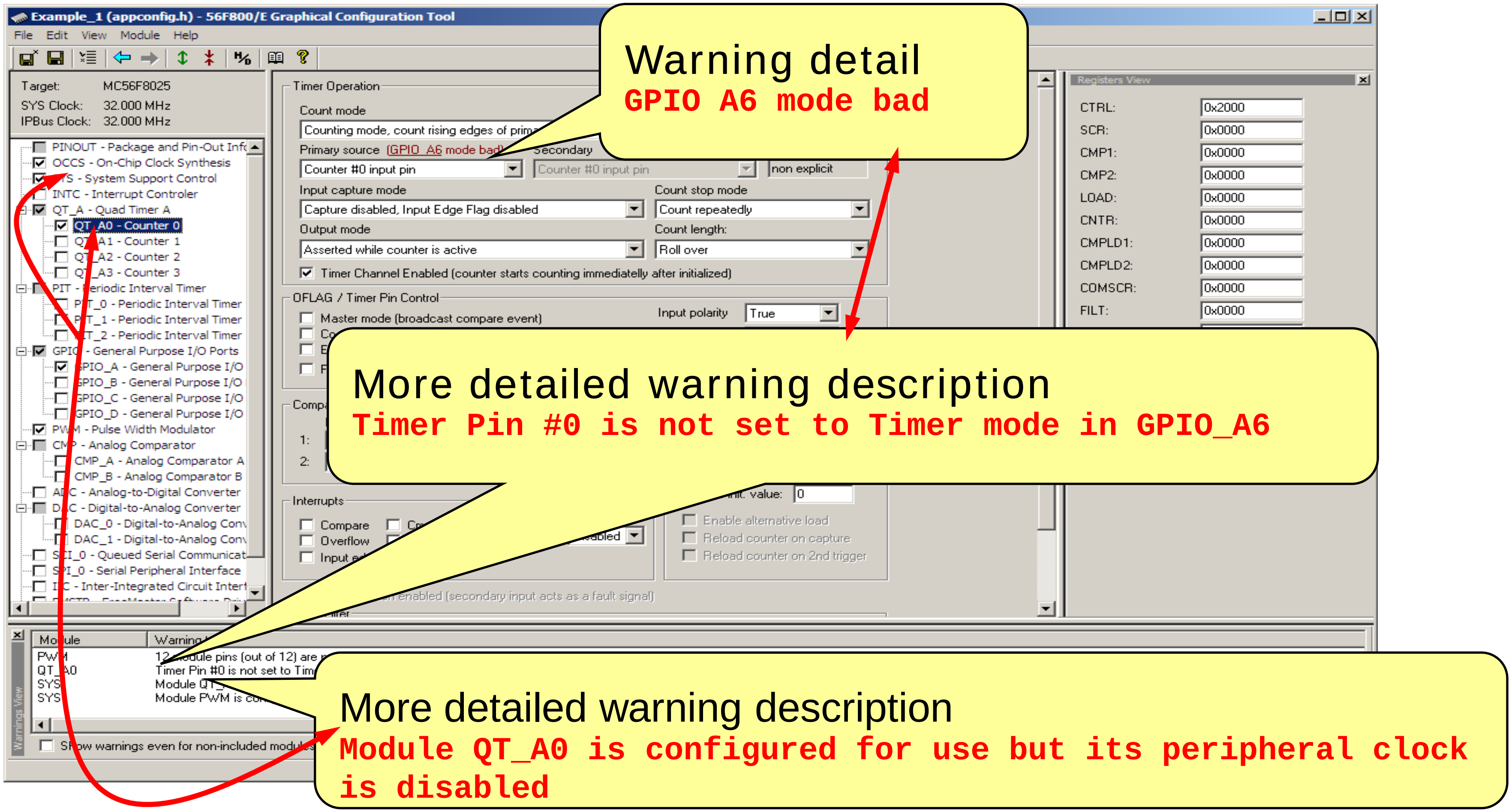
Module	Warning text
PWM	12 module pins (out of 12) are not available on any device pin. See GPIO_A and GPIO_B
SYS	Module PWM is configured for use but its peripheral clock is disabled

□ Show warnings even for non-included modules

NUM

Graphical Configuration Tool

- **Conflict Warnings**



Example_1 (appconfig.h) - 56F800/E Graphical Configuration Tool

File Edit View Module Help

Target: MC56F8025
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_A - Quad Timer A
  - ☑ QT_A0 - Counter 0
  - □ QT_A1 - Counter 1
  - □ QT_A2 - Counter 2
  - □ QT_A3 - Counter 3
- □ 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
  - □ DAC_1 - Digital-to-Analog Conv
- □ SCI_0 - Queued Serial Communicat
- □ SPI_0 - Serial Peripheral Interface
- □ IIC - Inter-Integrated Circuit Interf

Timer Operation

Count mode: Counting mode, count rising edges of prima

Primary source (GPIO_A6 mode bad): Counter #0 input pin

Secondary: Counter #0 input pin | non explicit

Input capture mode: Capture disabled, Input Edge Flag disabled

Count stop mode: Count repeatedly

Output mode: Asserted while counter is active

Count length: Roll over

☑ Timer Channel Enabled (counter starts counting immediatelly after initialized)

OFLAG / Timer Pin Control

□ Master mode (broadcast compare event)

Input polarity: True

Compa
1:
2:

Init. value: 0

Interrupts

□ Compare
□ Overflow
□ Input ed

□ Enable alternative load
□ Reload counter on capture
□ Reload counter on 2nd trigger

enabled (secondary input acts as a fault signal)

Registers View

CTRL:	0x2000
SCR:	0x0000
CMP1:	0x0000
CMP2:	0x0000
LOAD:	0x0000
CNTR:	0x0000
CMPLD1:	0x0000
CMPLD2:	0x0000
COMSCR:	0x0000
FILT:	0x0000

Warnings View

Module	Warning
PWM	12 module pins (out of 12) are n
QT_A0	Timer Pin #0 is not set to Tim
SYS	Module QT_
SYS	Module PWM is con

□ Show warnings even for non-included modules

Warning detail
GPIO A6 mode bad

More detailed warning description
Timer Pin #0 is not set to Timer mode in GPIO_A6

More detailed warning description
Module QT_A0 is configured for use but its peripheral clock is disabled

Quick Start Highlights

- **Highlights**
  - Quick_Start helps users to get familiar with the processor quickly
    - GCT helps to understand individual bits of peripheral registers
    - Sample applications demonstrate how to access the peripheral modules
  - Quick_Start helps users to jump in the SW development quickly
    - A ready-to-use project stationery to start a new project
    - GCT immediately available
  - No performance penalty when using Quick_Start
    - Optimal code, each instruction matters
    - Suitable for hard real-time applications (motor control)
    - Source files available, everything under control, no hidden code
- **Quality**
  - Developed under CMM-Level 3 certified process

FreeMASTER Tool

What is FreeMASTER?

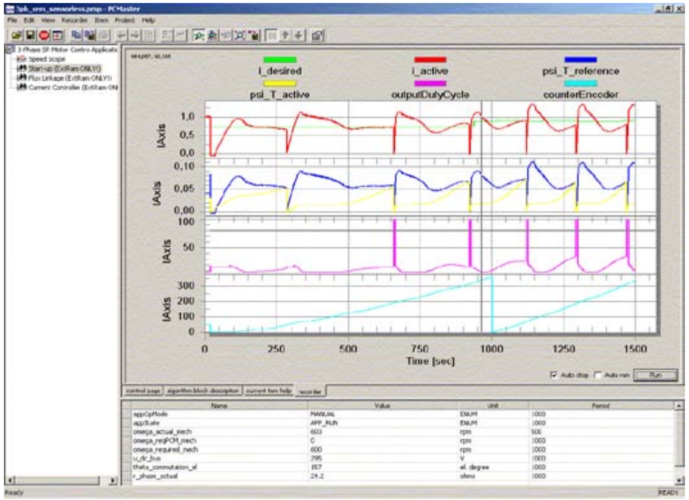
What is FreeMASTER?

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

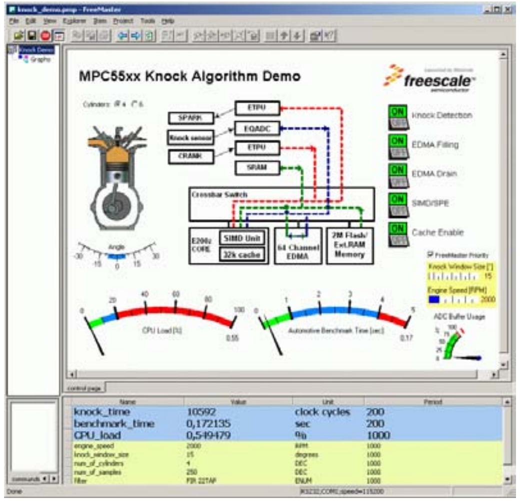
FOR YOUR

EMBEDDED

APPLICATION

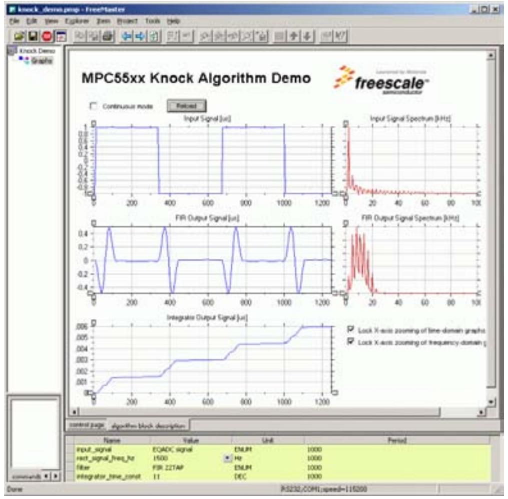


Name	Value	Unit	Period
appCpMode	MANUAL	ENUM	1000
appState	APP_RUN	ENUM	1000
omega_actual_mech	603	rpm	500
omega_reqPCM_mech	0	rpm	1000
omega_required_mech	600	rpm	1000
u_dc_bus	295	V	1000
theta_commutation_el	157	el. degree	1000
r_phase_actual	24.2	ohms	1000



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



MPC55xx Knock Algorithm Demo
freescale semiconductor
Continuous mode
Reload
Input Signal [u]
Input Signal Spectrum [kHz]
FIR Output Signal [u]
FIR Output Signal Spectrum [kHz]
Integrator Output Signal [u]
Lock X-axis zooming of time-domain graphs
Lock X-axis zooming of frequency-domain graphs
control page
algorithm block description

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 OVERVIEW

FreeMASTER as a Real-Time Monitor

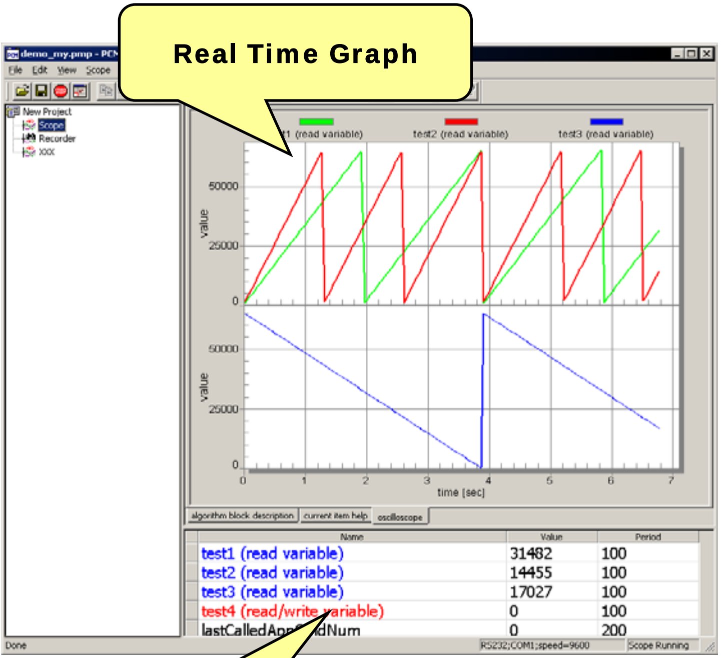
FreeMASTER as a Real-time Monitor

- Connects to an embedded application
  - SCI, UART
  - JTAG/EOnCE (56F8xxx only)
  - BDM (HCS08, HCS12 only)
  - CAN Calibration 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

FreeMASTER as a Real-time Monitor

- Displays the variable values in various 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**



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

Real Time Graph

Variable Watch

FreeMASTER as a Real-Time Monitor

Additional features:

- Variable Transformations
  - Variable value can be transformed to custom unit
  - Variable transformations may reference other variable values
  - Values are transformed back when writing a new value to 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 PC
- Ability to protect memory regions
  - Describing variables visible to FreeMASTER
  - Declaring variables as read-write to read-only for FreeMASTER
    - the 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 (e.g. motor control)
- Recorder may be used to visualize transitions in near 10-us resolution

FreeMASTER

as a Graphical User Interface to the Embedded Application

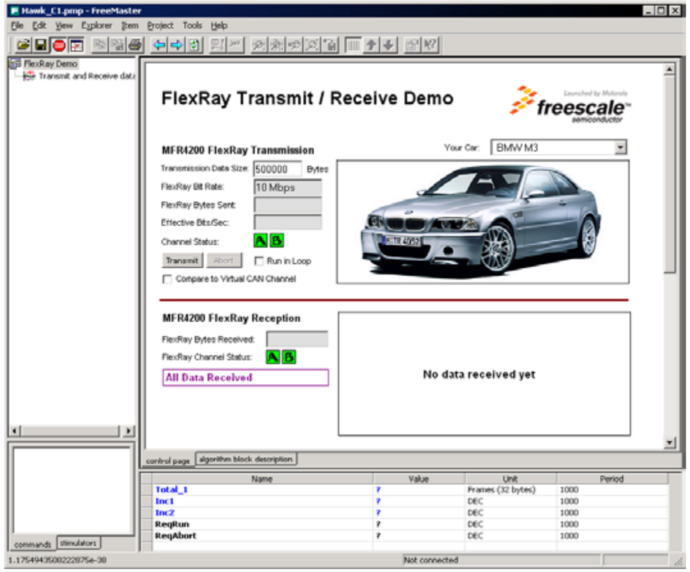
FREEMASTER OVERVIEW

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
freescale semiconductor
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

Transmit Abort □ 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

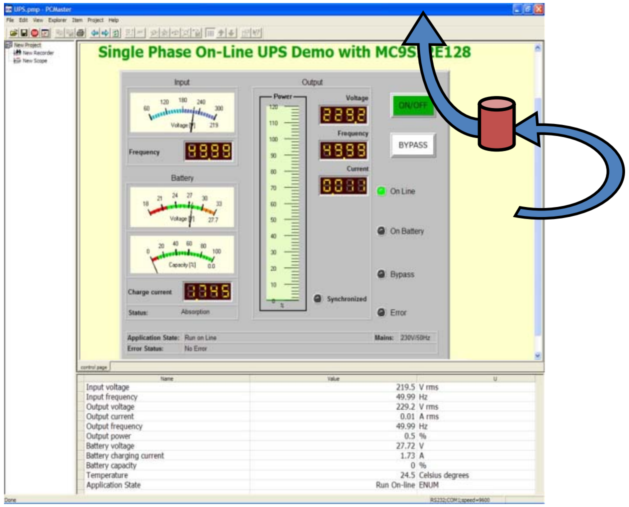
1.175494350822875e-38

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



Single Phase On-Line UPS Demo with MC9S E128

Status: Absorption

Application State: Run on Line

Error Status: No Error

Mains: 230V/50Hz

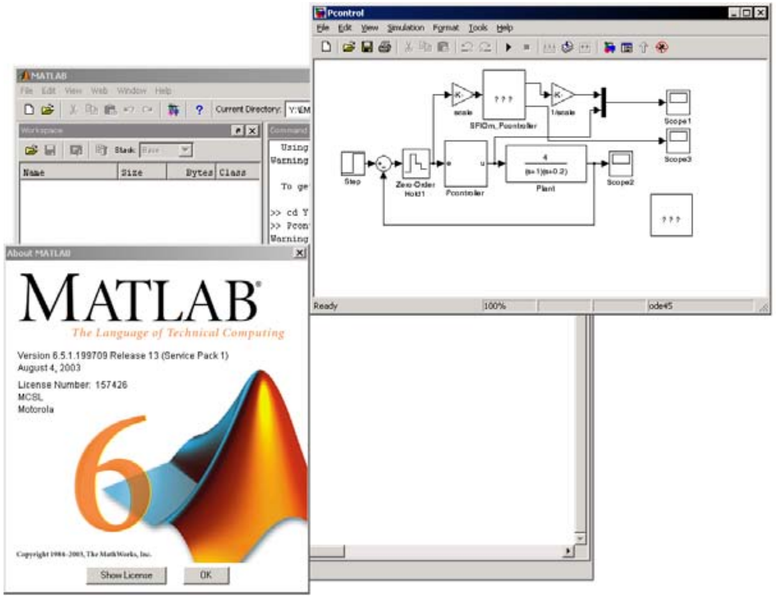
Name	Value	U
Input voltage	219.5	V rms
Input frequency	49.99	Hz
Output voltage	229.2	V rms
Output current	0.01	A rms
Output frequency	49.99	Hz
Output power	0.5	%
Battery voltage	27.72	V
Battery charging current	1.73	A
Battery capacity	0	%
Temperature	24.5	Celsius degrees
Application State	Run On-line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

- 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



MATLAB

The Language of Technical Computing

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

License Number: 157426
MCSL
Motorola

Copyright 1984-2003, The MathWorks, Inc.

```
Using
Warning

To ge

>> cd Y
>> Pcon
Warning
```

FreeMASTER

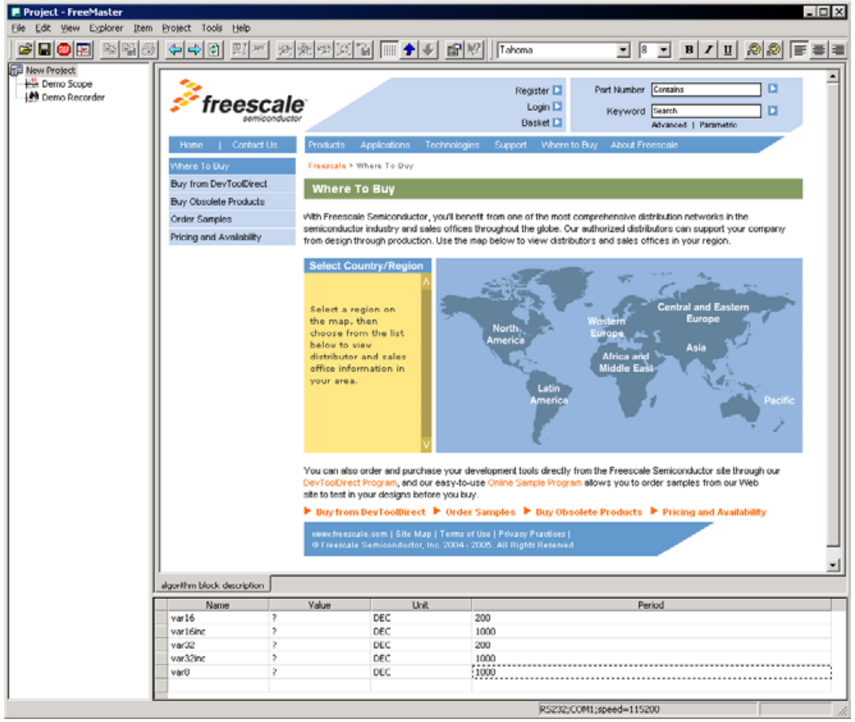
as a Demonstration Platform & Selling Tool

FREEMASTER OVERVIEW

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&fsrch=1

Thank you