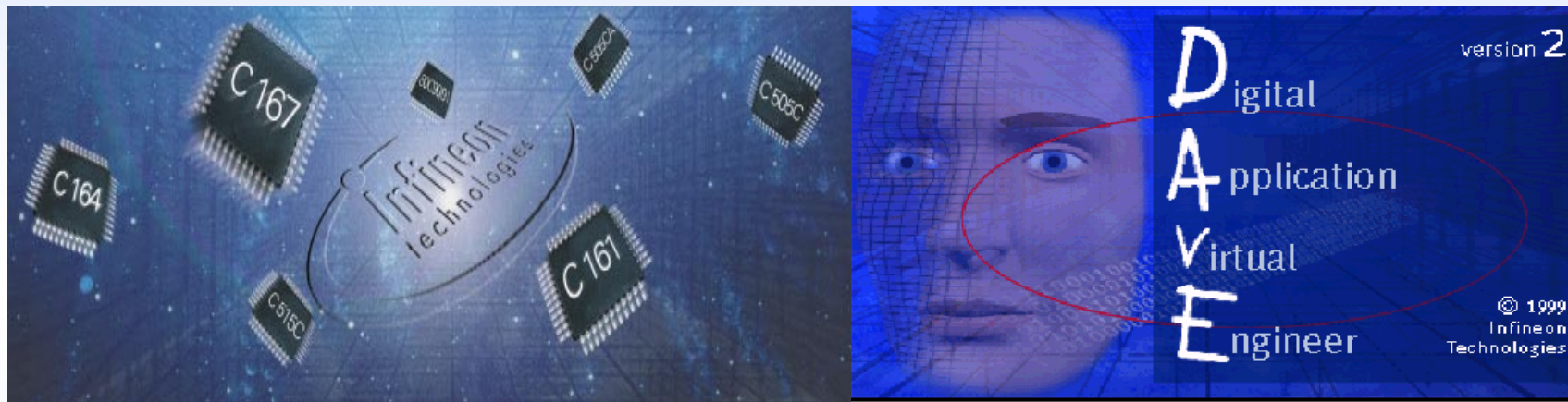


# HOT504\_1

## A Hands-On Introduction to the C504

using DAVe 2.0, the C504 Starter Kit,  
Keil  $\mu$ Vision + dScope, and an oscilloscope



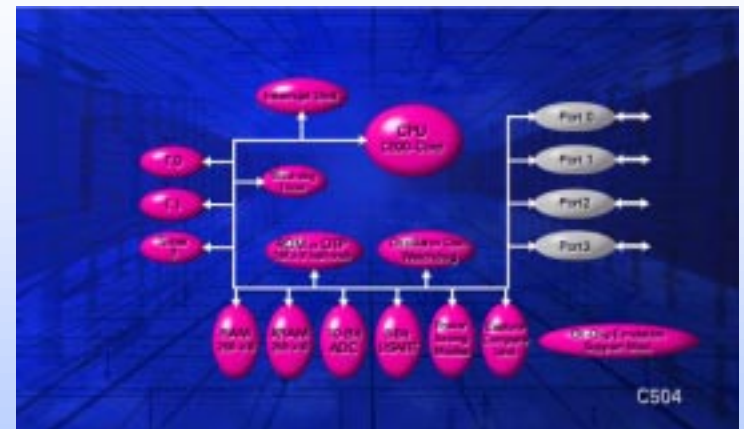
This version is based on DAVe Version 2.0, Keil  $\mu$ Vision V1.24,  
dScope V1.3w, Starter Kit CD ROM V3.0.

Please report any errors to [mike.copeland@infineon.com](mailto:mike.copeland@infineon.com)

# Contents (I)

## □ Introduction

- Introduction to HOT504\_1
- Short Introduction to DAvE 2.0
- Short Introduction to Keil  $\mu$ Vision



# Contents (II)

---

## □ C500 Architectural Overview

- 8 bit Products
- Product Numbering Scheme
- C504 Block Diagram
- C500 CPU Core
- Internal & External Memory
- Interrupt System
- C504 Peripheral Overview



# Contents (III)

---

## ❑ C504 Hands-On Peripheral Training

- Exercise Overview
- How to set up the Hardware and Software
- Files to build a program
- Hints regarding DAVe 2.0 and the Exercises
- Peripherals in Detail + Exercises



# Introduction to HOT504\_1

---

- ❑ **HOT504\_1 is a Hands-On Training material created for the C504, using**
  - the KitCON-504 Starter Kit
  - the Keil uVision development environment including the C51 compiler, A51 Assembler, BL51 Linker/Locator
  - the Keil dScope Debugger
  - DAvE, the Digital Application Engineer from Infineon Microcontrollers (Version 2.0)
  - an oscilloscope (for visualization purposes).
  - A Windows95 or Windows NT PC
- ❑ **HOT504\_1 shows the user from scratch how to generate software for the C504 with DAvE and the Keil tool chain:**
  - There are several exercises included, small tasks to be solved using every peripheral of the C504.
  - The user creates a new project in DAvE and configures the device, following the detailed instructions.

# Introduction to HOT504\_1 (cont.)

---

- After having generated the code, the user
  - switches to Keil  $\mu$ Vision,
  - creates a new project,
  - includes the C files created by DAvE,
  - adds some User Code,
  - compiles, assembles, links and locates the project.
- After compilation with  $\mu$ Vision, the user
  - switches to the dScope Debugger
  - connects to the kitCON-504 via the PC serial port,
  - loads, starts and debugs his example,
  - confirms his working program with a scope (screen shots are included for most of the examples).

# Short Introduction to DAVe 2.0

---

- ❑ **DAVe is your Digital Application Engineer from Infineon Microcontrollers.**
- ❑ **DAVe can help you compare and evaluate the different members of the Infineon C500 (8-Bit) and C166 (16-Bit) families of microcontrollers and help you find the right chip for your embedded control application.**
- ❑ **DAVe can be your one-stop access point to all standard knowledge associated with Infineon embedded technology expertise by offering you context sensitive access to user's manuals, data sheets, application notes etc. directly in your development environment.**
- ❑ **DAVe can help you program the Infineon microcontroller you want to use in your project, by offering you intelligent wizards that help you configure the chip to work the way you need it and automatically generate C-level templates with appropriate access functions for all of the on chip peripherals and interrupt controls.**
- ❑ **You can follow an exciting online multimedia tutorial on how to use DAVe. To start the tutorial click on the "Tutorial" item in the help menu.**
- ❑ **More DAVe info at [www.infineon.com/DAVe](http://www.infineon.com/DAVe)**

# Differences between DAVe 1.0 and 2.0

---

- DAvE 1.0 was CD-ROM based. In DAVe 2.0 you need the CD-ROM during the installation procedure only. In the setup program you have to specify the controllers you want to work with.**
- In DAVe 1.0 you got a context sensitive menu for each peripheral which offered you a short description, a link to the User's Manual or to start the Configuration wizard. In DAVe 2.0 the short description is not available any more. The link to the User's Manual is available in the Configuration Wizard now. To start the Configuration Wizard click on the highlighted peripheral.**
- In DAVe 1.0 you could open only one Configuration Wizard at the same time. In DAVe 2.0 you can open several Configuration Wizards at the same time.**



## Differences between DAvE 1.0 and 2.0 (cont.)

---

- In DAvE 1.0 your changes in a Configuration Wizard would be saved to disk when the OK Button would be clicked. In DAvE 2.0 changes in the Configuration Wizard will be saved immediately in the internal memory. These changes will be saved to disk when you save them manually by clicking the "Save" item in the "File" menu.
- In DAvE 2.0 you can generate code even if you have a Configuration wizard opened.
- The context sensitive menus in the Configuration Wizards are not available any more. To link to the User's Manual click on the information/documentation button (second one from left) to get some items which link to the User's Manual. To get information why a setting is disabled or to get short descriptions for functions and macros watch the tool tips for the specific controls. Note that you got detailed function and macro descriptions in the project documentation and in the generated code.

# Short Introduction to the Keil $\mu$ Vision Integrated Development Environment

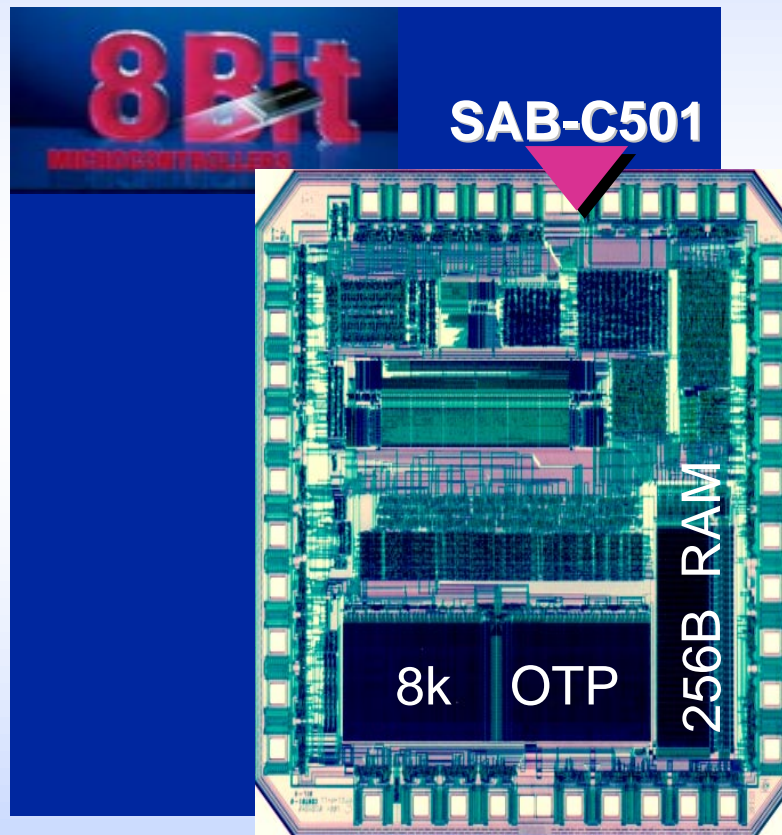
---

## ❑ Keil $\mu$ Vision + dScope:

- $\mu$ Vision, the IDE from Keil Software, combines Project Management, Source Code Editing, and Program Debugging in one powerful environment. The Quick Start guide on the starter Kit CD ROM gives you the information necessary to use  $\mu$ Vision2 for your own projects. It provides a step-by-step introduction of the most commonly used  $\mu$ Vision2 features including:
  - Project Setup for the Make and Build Process
  - Editor facilities for Modifying and Correcting Source Code
  - Program Debugging and Additional Test Utilities

## ❑ More information is available on the Starter Kit CD ROM or at [www.keil.com](http://www.keil.com).

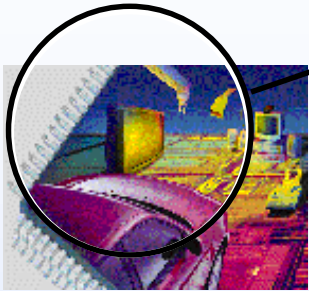
# C500 Architectural Overview



- 8 bit Products
- Product Numbering Scheme
- C504 Block Diagram
- C500 CPU Core
- Internal & External Memory
- Interrupt System
- C504 Peripheral Overview

# 8-bit Products

## High-Integration



### C509

- \* >3KB RAM
- \* MDU
- \* 512 Byte Boot ROM
- \* Many Timers
- \* USART
- \* 29 ch. Capture/Compare Unit
- \* 8 data-pointers
- \* P-MQFP-100

### C505L

- C505 with
- \* 32k OTP
- \* 128 seg. LCD controller
- \* Real Time Clock
- \* 8 ch. 10-bit A/D
- \* P-MQFP-80

### C515C

- C515A with
- \* 64K ROM/OTP
- \* +1KB RAM
- \* Full CAN 2.0B Active Controller
- \* SSC
- \* 8 ch. 10 bit A/D
- \* 8 data-pointers

### C541U

- \* 8K OTP
- \* 256 Bytes RAM
- \* Full & Low Speed USB Module 4+1 EP
- \* SSC
- \* 2 16 bit Timers
- \* P-LCC-44

### C517A

- Replaces 80C517A
- \* >2k RAM
- \* 32k ROM
- \* USART
- \* UART
- \* 21 ch Capture/Compare Unit
- \* 12 ch. 10 bit A/D
- \* Many Timers
- \* 8 Data Pointers
- \* P-LCC-84

### C504

- \*16K ROM/OTP
- \* 512 Bytes RAM
- \* 8 ch. 10-bit A/D
- \* Motor Control Peripheral
- \* 3 16-bit Timers
- \* USART

### C505CA

- \* 32k ROM or OTP
- \* 1280 bytes RAM
- \* Full CAN 2.0B Active Controller
- \* 4 ch. Capture/Compare Unit
- \* 8 ch. 10 bit A/D
- \* USART
- \* 3 16-bit Timers
- \* 8 data-pointers

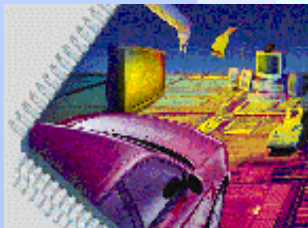
### 80C515A

- C515 with:
- \* 32K ROM
- \* + 1kB RAM
- \* 8 ch. 10 bit A/D
- \* P-LCC-68

### C508

- \* 1280 bytes RAM
- \* 32k OTP/ROM
- \* USART
- \* Enhanced Motor Control Peripheral
- \* 8 ch. 10 bit A/D
- \* Extra 4 ch. Capture/Compare Unit
- \* 8 Data Pointers
- \* Built-in PLL (x2)
- \* P-MQFP-64

## General Purpose



### C515

- \* 8 k ROM
- \* 256 bytes RAM
- \* 4 ch. Capture / Compare Unit
- \* 8 ch. 8 bit A/D
- \* USART
- \* 3 16-bit Timers
- \* PLCC 68

### C505A

- \* 32k ROM or OTP
- \* 1280 bytes RAM
- \* 4 ch. Capture/Compare Unit
- \* 8 ch. 10 bit A/D
- \* USART
- \* 3 16-bit Timers
- \* 8 data-pointers

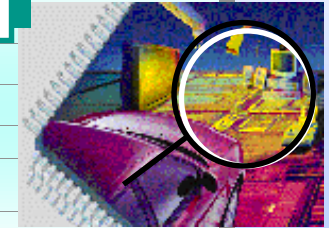
### C501G-1E

- \* 8k OTP
- \* 3 16-bit Timers
- \* USART
- \* P-MQFP-44
- \* P-LCC-44
- \* P-DIP-40

### C513AO

- \* 512 Bytes RAM
- \* 16 k ROM/OTP
- \* USART
- \* SSC
- \* 3 16-bit Timers
- \* P-LCC-44

## Low-Cost



# C504 Part Numbering Scheme

SA

F

C504

—

-

2

E

24

M

**Standard Prefix:**  
Always "SA"

**Temperature Range:**  
B = 0 to 70°C    H = -40 to 110°C  
F = -40 to 85°C    K = -40 to 125°C

**Basic Type (with CMOS indicator):**  
One letter with three numbers  
e.g. C505, C167

**Functionality Option Field (optional):**  
blank for the C504

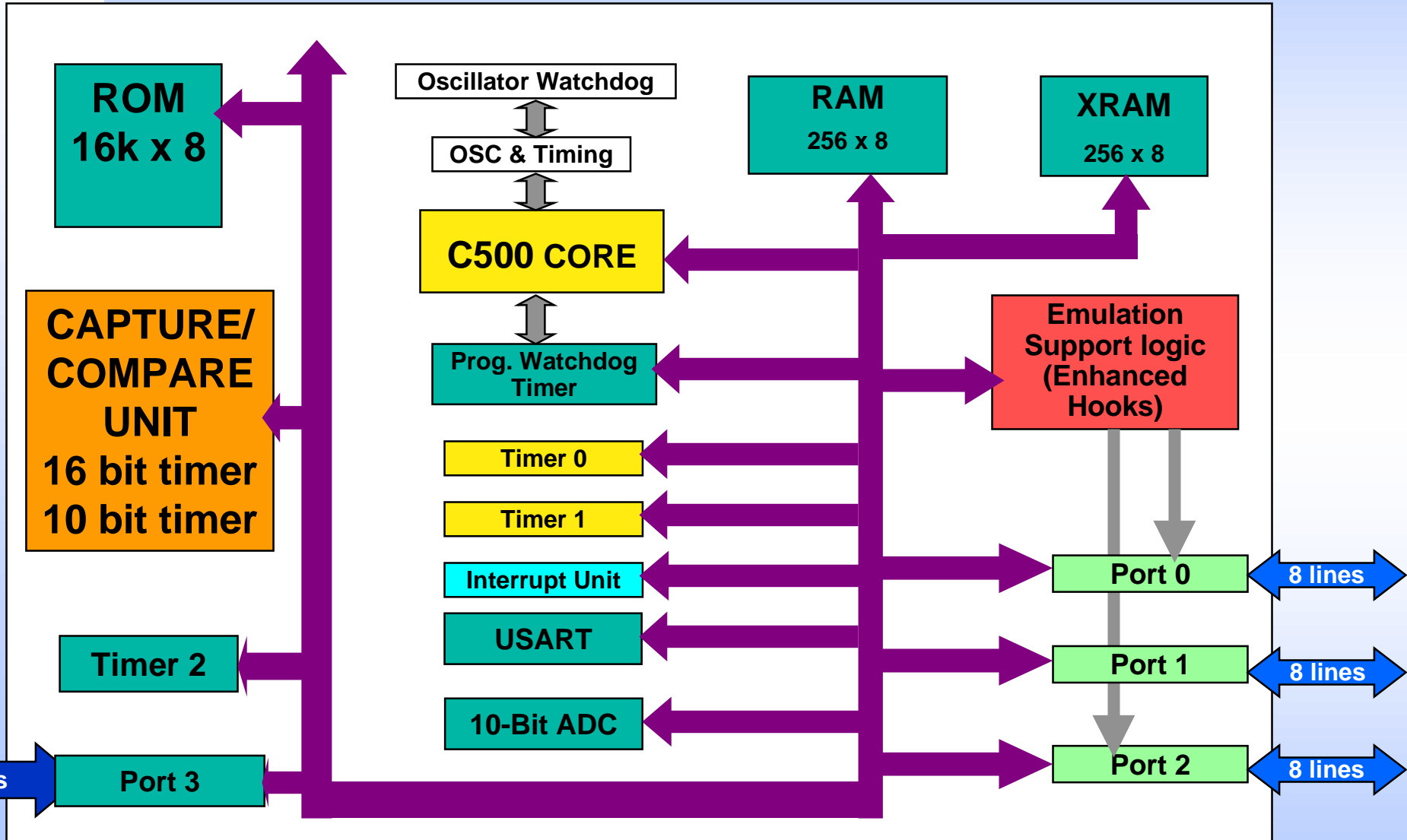
**ROM Size (n x 8k):**  
(blank) = ROMless  
1 = 8k    4 = 32k  
2 = 16k    8 = 64k

**Package Type:**  
P = P-DIP  
M = P-MQFP  
N = P-LCC

**Speed Designator:**  
(blank) = 12 MHz  
24 = 24 MHz  
40 = 40 MHz

**ROM Type:**  
L = ROMless  
R = Mask ROM  
E = EPROM (OTP)  
F = Flash  
H = EEPROM

# C504-2RM Block Diagram



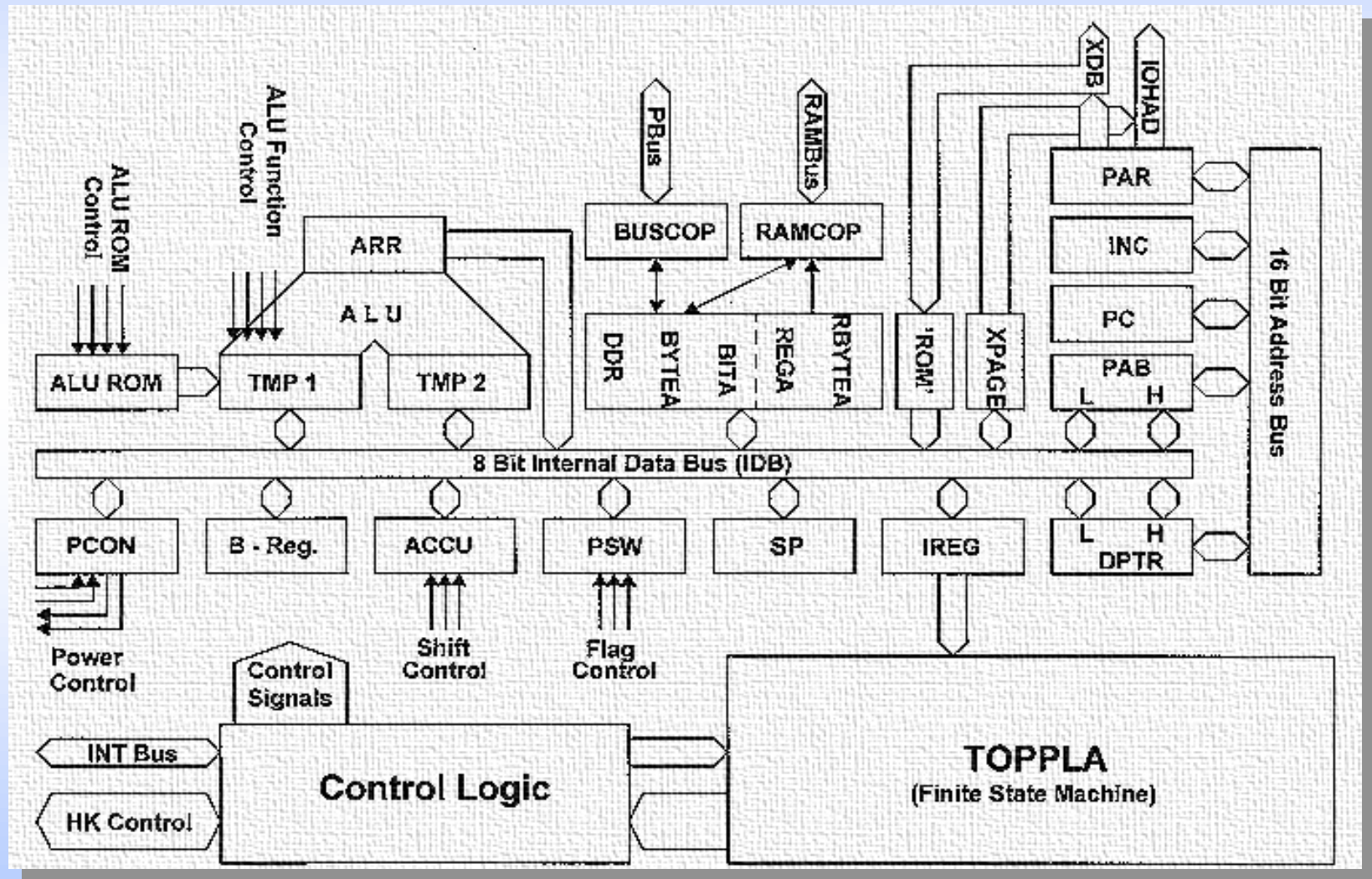
# Architectural Overview

## (adapted to C504)

---

- ❑ **Complete 8-bit architecture with fully compatible instruction set to standard 8051 microcontroller**
- ❑ **Up to 40 MHz CPU clock results in an instruction cycle time of 300ns which guarantees highest CPU performance**
- ❑ **To avoid an accumulator bottleneck four banks with eight General Purpose Registers (GPRs) are implemented**
  - The register banks are fixed located in the lower 32 locations of the internal RAM
- ❑ **Easy and efficient programming is supported by powerful instructions**
  - 64 of 111 instructions are executed in one machine cycle
  - 256 directly addressable bits
- ❑ **Transparent programming of on-chip peripherals via Special Function Register (SFR) interface**

# Block Diagram C500 CPU (8051 Compatible)





# Addressing Modes

---

## ❑ Register Addressing

- 8 registers (R0 - R7) in one of the four available register banks
  - e.g. MOV A,Rr

## ❑ Direct Addressing

- Special Function Registers (SFRs) and the lower 128 bytes of internal RAM
  - e.g. MOV dadr1,dadr2

## ❑ Immediate Addressing

- Constants in the program memory are allowed to be part of the instruction
  - e.g. MOV A,#const8

# Addressing Modes (cont.)

---

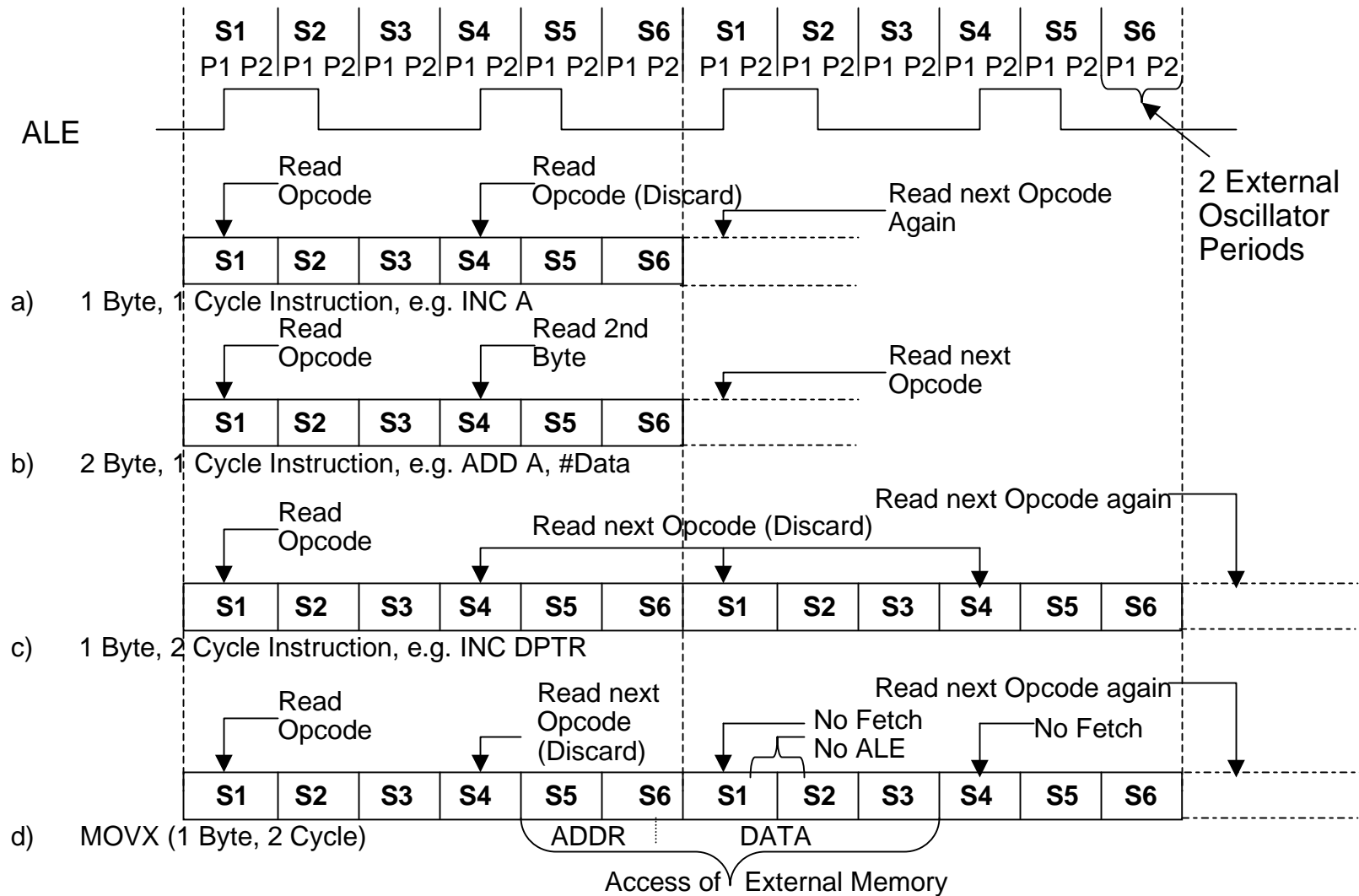
## ❑ Register Indirect Addressing

- 256 bytes of internal RAM or the lower 256 bytes of external data memory (or XRAM) via the contents of either R0 or R1 in the selected register bank
  - e.g. MOV @Ri,A or MOVX @Ri, A
- External data memory using 16 bit data pointer
  - e.g. MOVX @DPTR, A

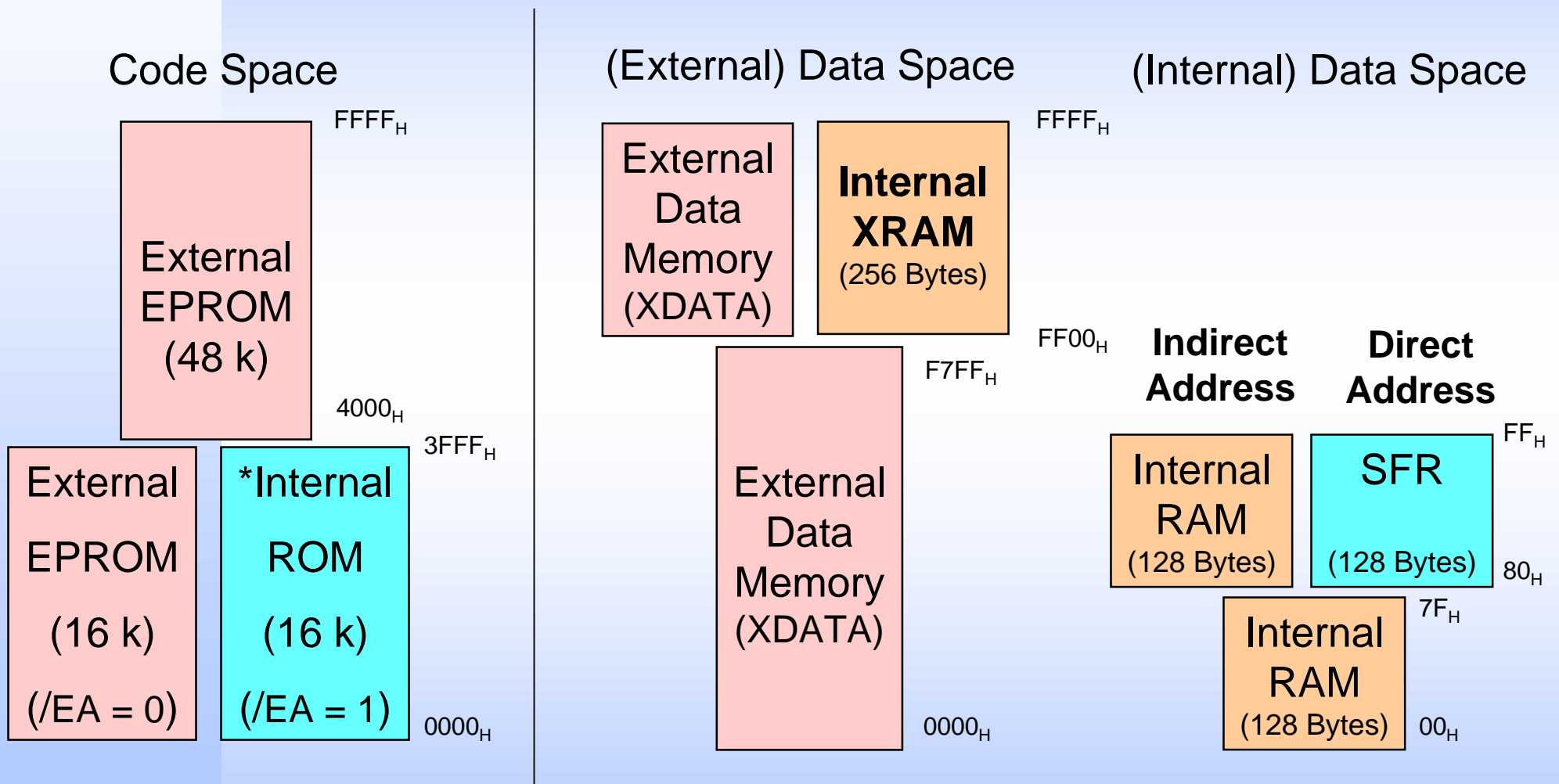
## ❑ Base Register Plus Index Register Addressing

- Indirect move from the location whose address is the sum of a base register (DPTR or PC) and index register
  - e.g. MOVC A, @A+DPTR

# CPU Fetch/Execute Sequence



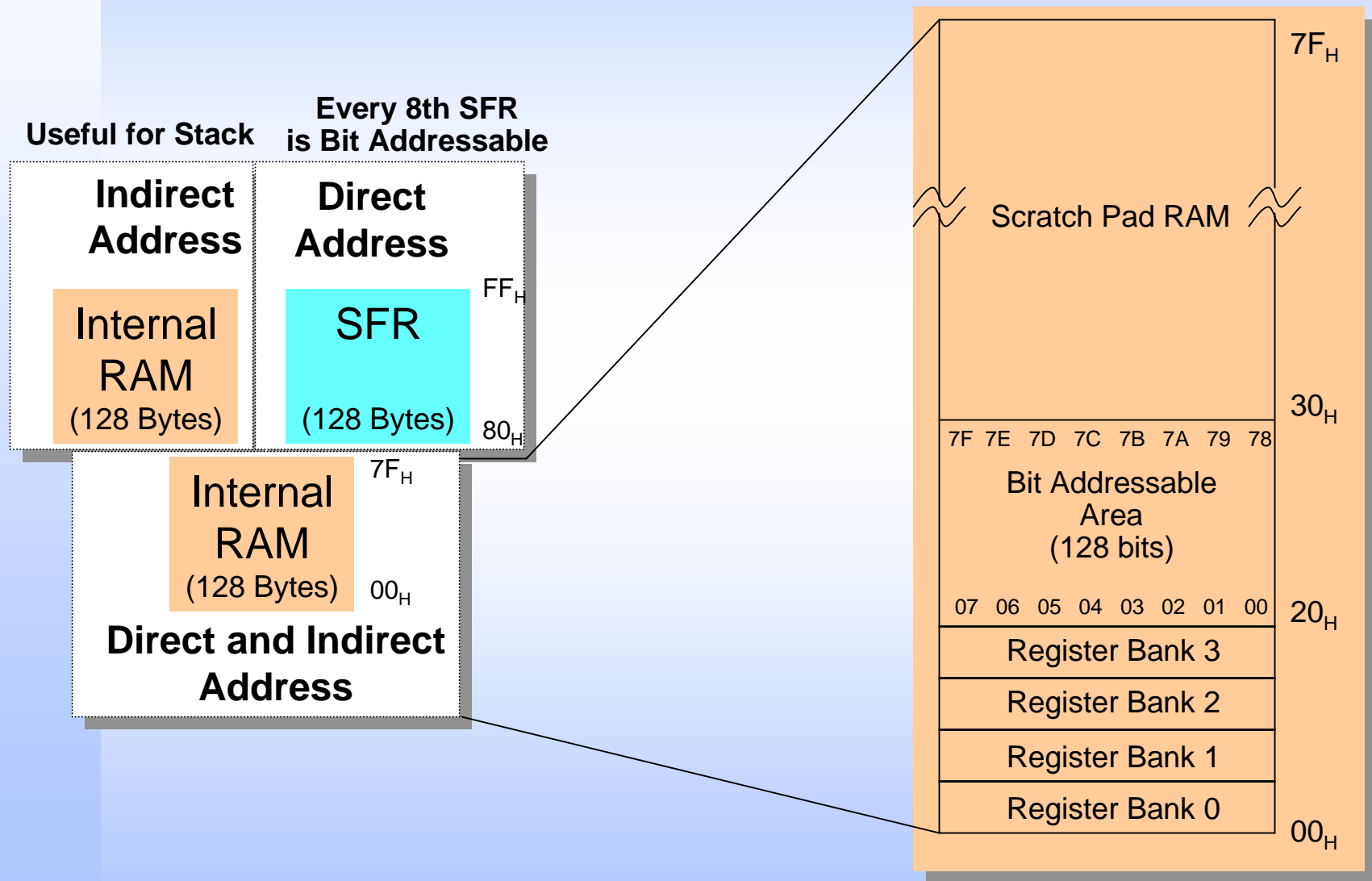
# Memory Map -- C504-2RM (Harvard Architecture)



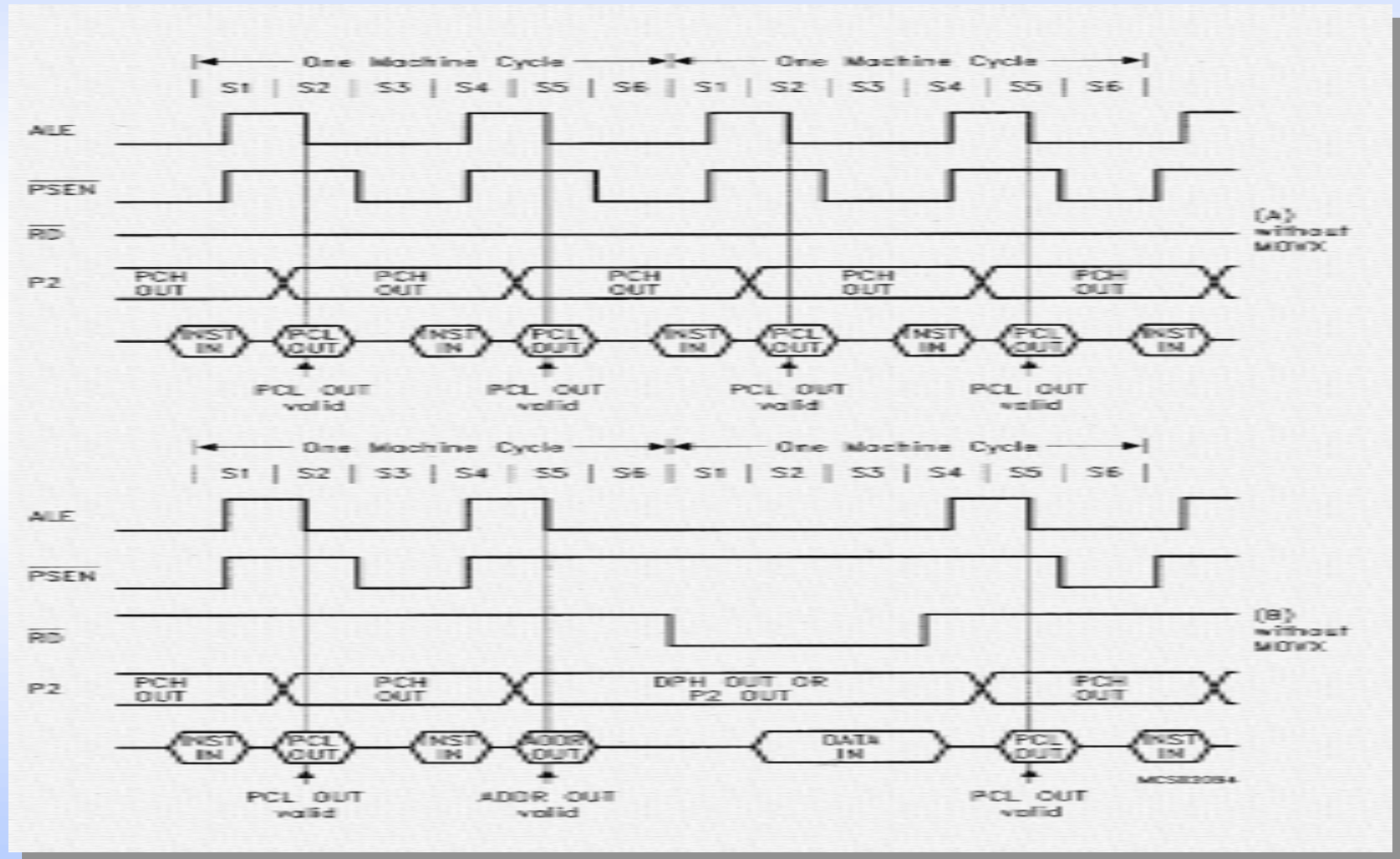
\* C5054-2RM has 16k ROM. C504-4EM has 16k OTP. C504-LM has no internal ROM

# Memory Map (cont.)

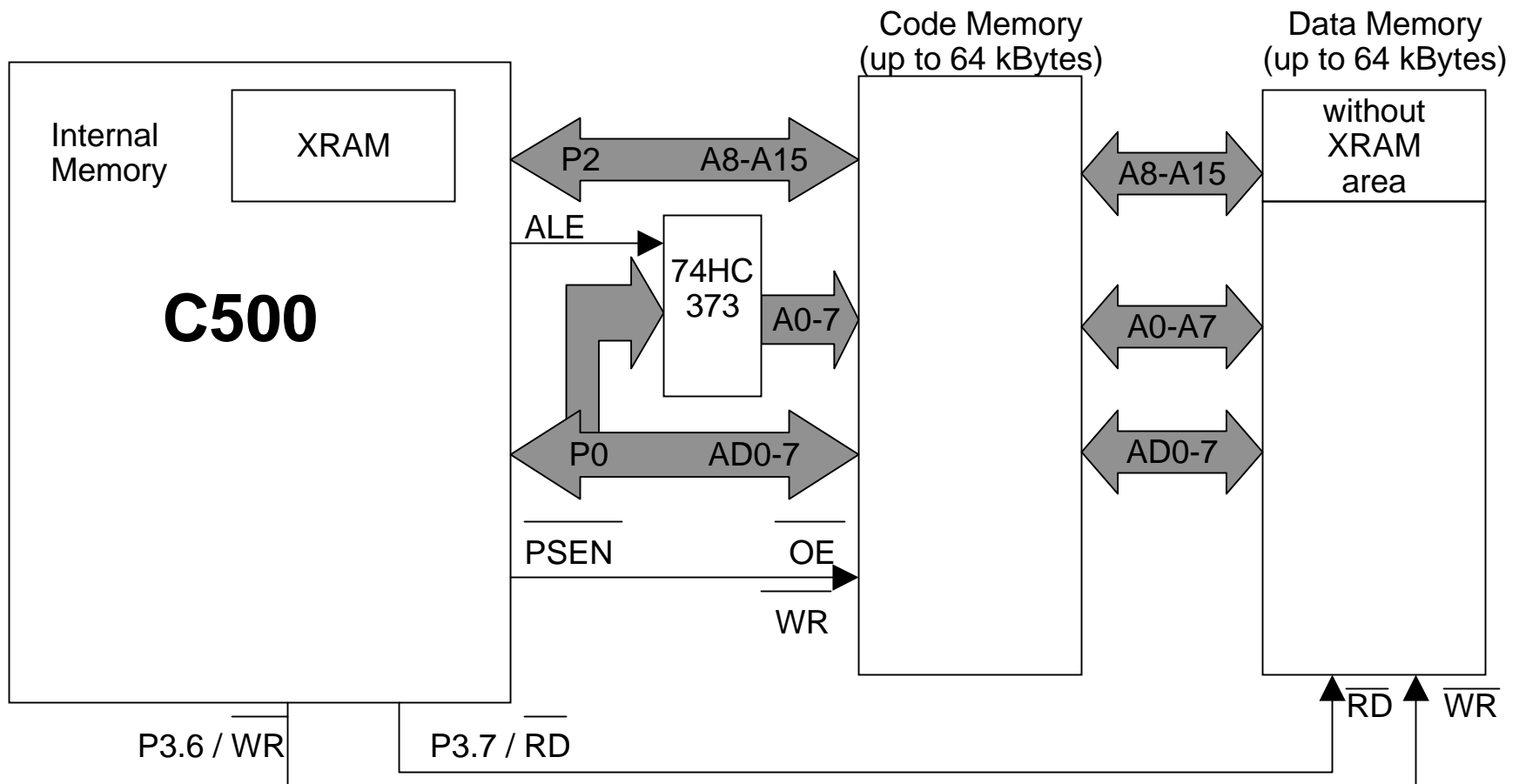
## Internal Data Space



# External Bus Access



# External Connection to Memory



# Interrupt System

---

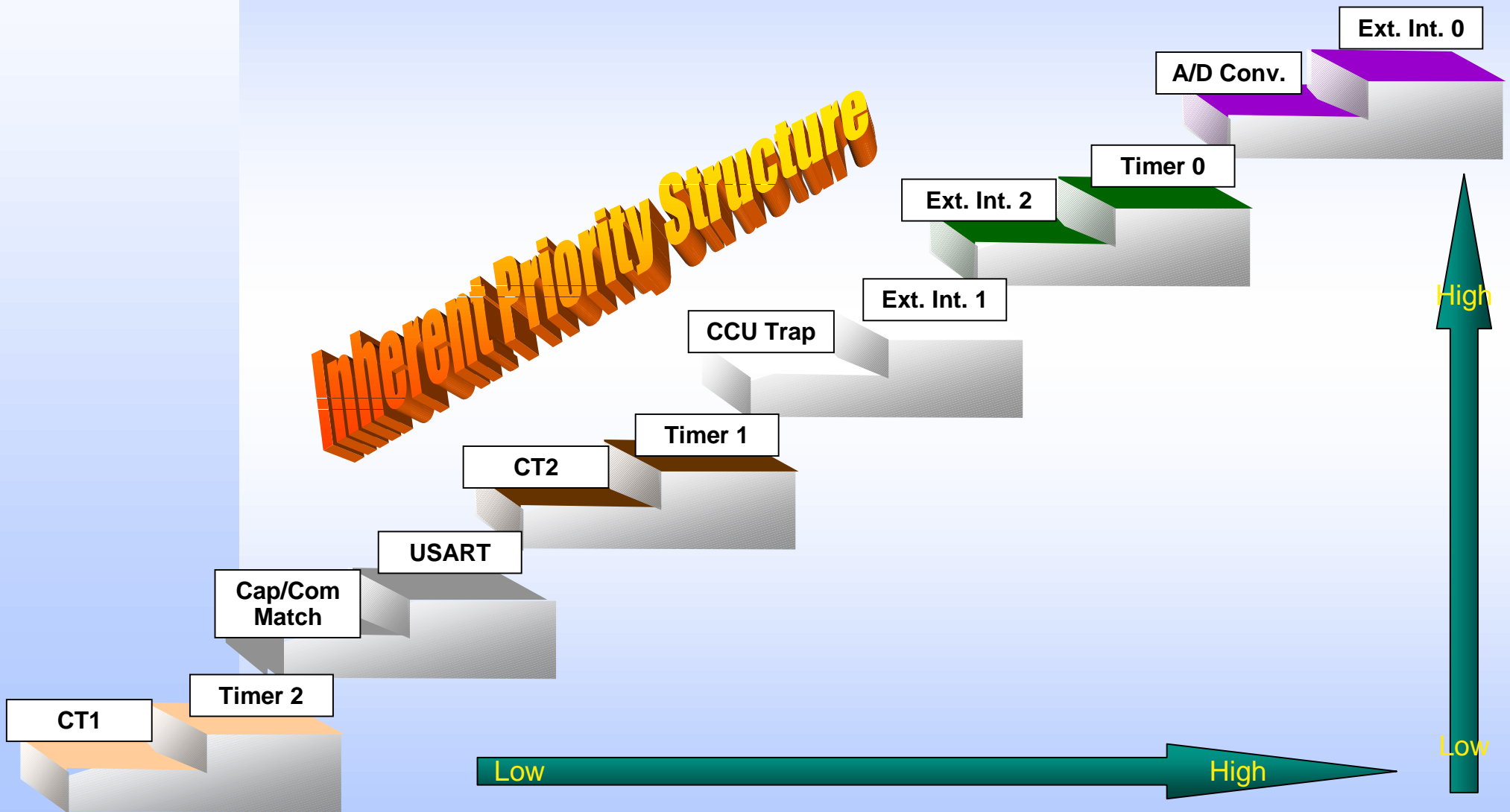
## □ Interrupt Controller

- Short interrupt response time
- Context Switching
- Short interrupt service overhead
- Low total latency gives highest real-time performance
- Comprehensive prioritization scheme
  - **Easy scheduling of complex real-time systems by using 2 priority levels**
  - **Simultaneous interrupts of same priority undergo a group-wise arbitration (using the inherent priority structure)**

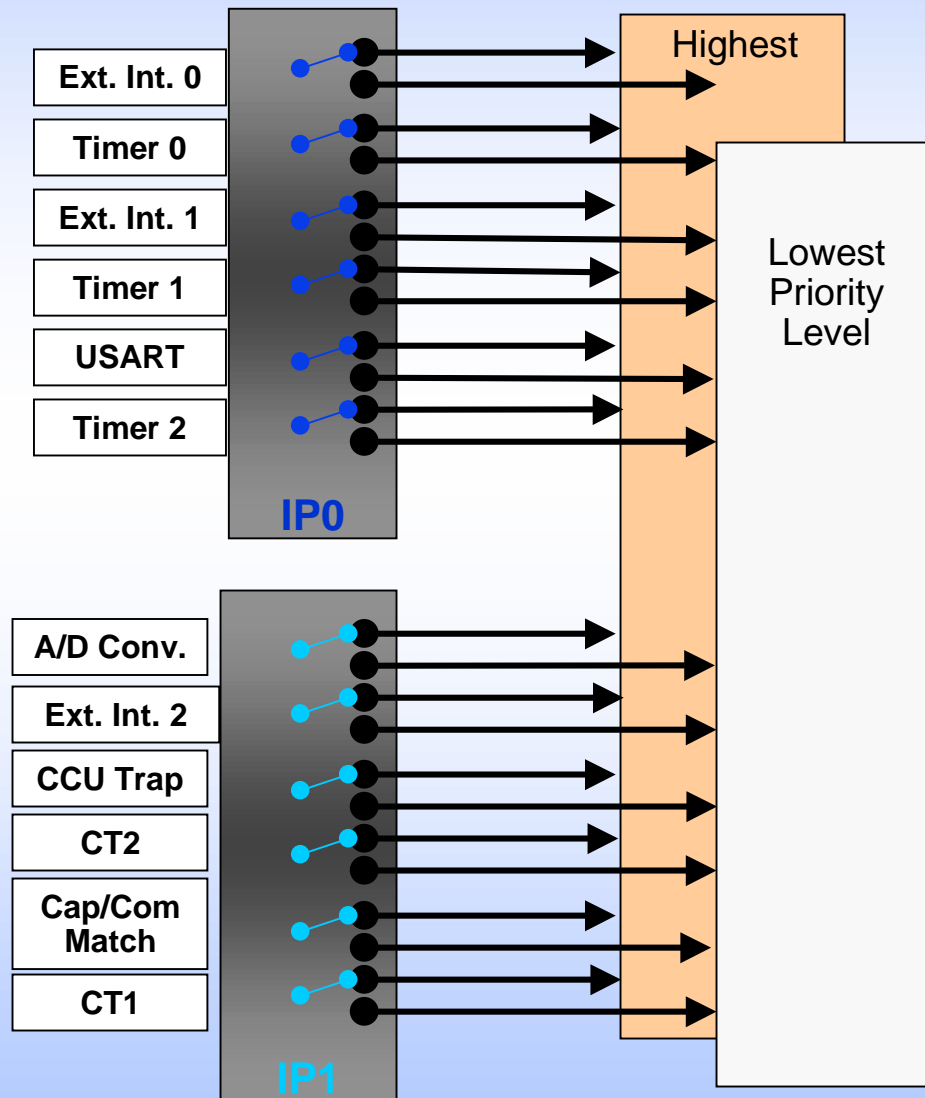


# Inherent Priority Structure

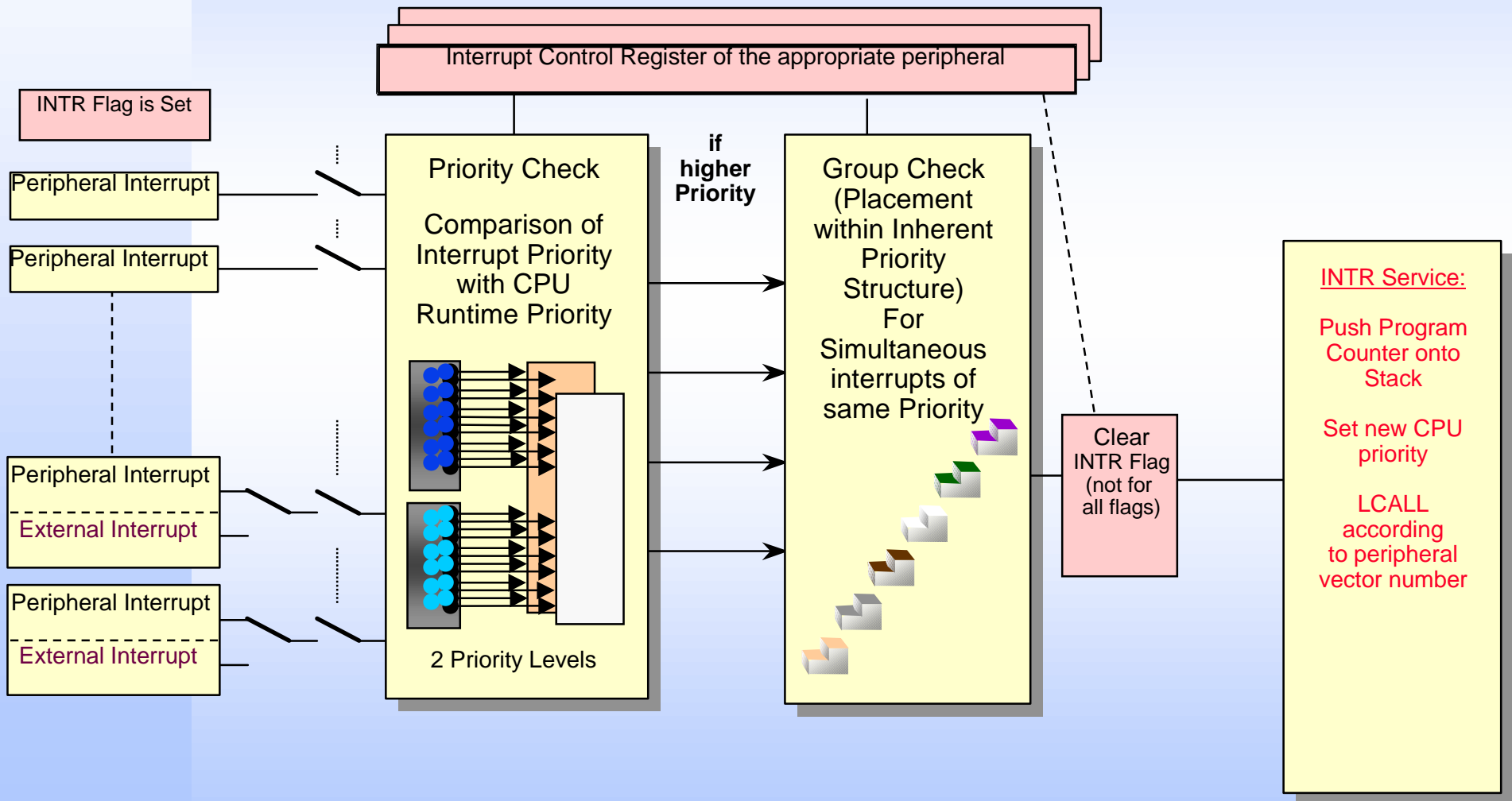
- Only used for Simultaneous Interrupts of Same Priority



# Two Programmable Priority Levels



# Interrupt Processing



# Peripherals Set of the C504

---

- ❑ **2 Timer/Counter units (T0 & T1)**
  - 16 bit timers with 4 operating modes (including 1 reload mode)
- ❑ **A 3rd 16 bit Timer/Counter**
  - 16 bit auto-reload (up or down counting)
  - 16 bit Capture Mode
  - Baud rate generator for USART
- ❑ **Independent Full Duplex USART**
  - 3 Asynchronous modes, 1 Synchronous mode
- ❑ **8 channel A/D Converter**
  - 10 bit resolution with 8 multiplexed inputs
- ❑ **Multi-functional Capture/Compare Unit**
  - 16 bit Timer and 10 bit Timer
  - General Purpose and Motor Control modes
- ❑ **Watchdog: 15-bit Reload-timer causes reset on overflow**

# C504 Hands-On Peripheral Training

---



- Exercise Overview
- How to set up the Hardware and Software
- Peripherals in Detail + Exercises
  - Timer/Counter 0
  - Timer/Counter 1
  - USART
  - A/D Converter
  - Timer/Counter 2
  - Capture/Compare Module

# Exercise Overview

---

## ❑ Exercises for USART and T0/T1:

- 4URT\_1 - Asynchronous Serial Transmission of data periodically
- 4URT\_2 - Synchronous Serial Transmission of data periodically

## ❑ Exercise for the Analog to Digital Converter and Timer 2:

- 4ADC\_1 - Generate a variable frequency square wave \*\*

## ❑ Exercises for the Capture / Compare Module:

- 4CCU\_1 - Symmetrical PWM Generation
- 4CCU\_2 - Synchronous Motor PWM Generation

\*\* Requires External Connections and Components

# Exercise Overview Table

	Peripherals						Functions		
	T0	T1	T2	USART	A/D	CAP/COM	ISR	DAvE Function Lib	
Exercises	4URT_1	X	X	-	X	-	-	X	X
	4URT_2	-	X	-	X	-	-	X	X
	4ADC_1	-	-	X	-	X	-	X	X
	4CCU_1	-	-	-	-	-	X	-	X
	4CCU_2	-	-	-	-	-	X	-	X

# How to set up your system: KitCON-504 HW setup

---

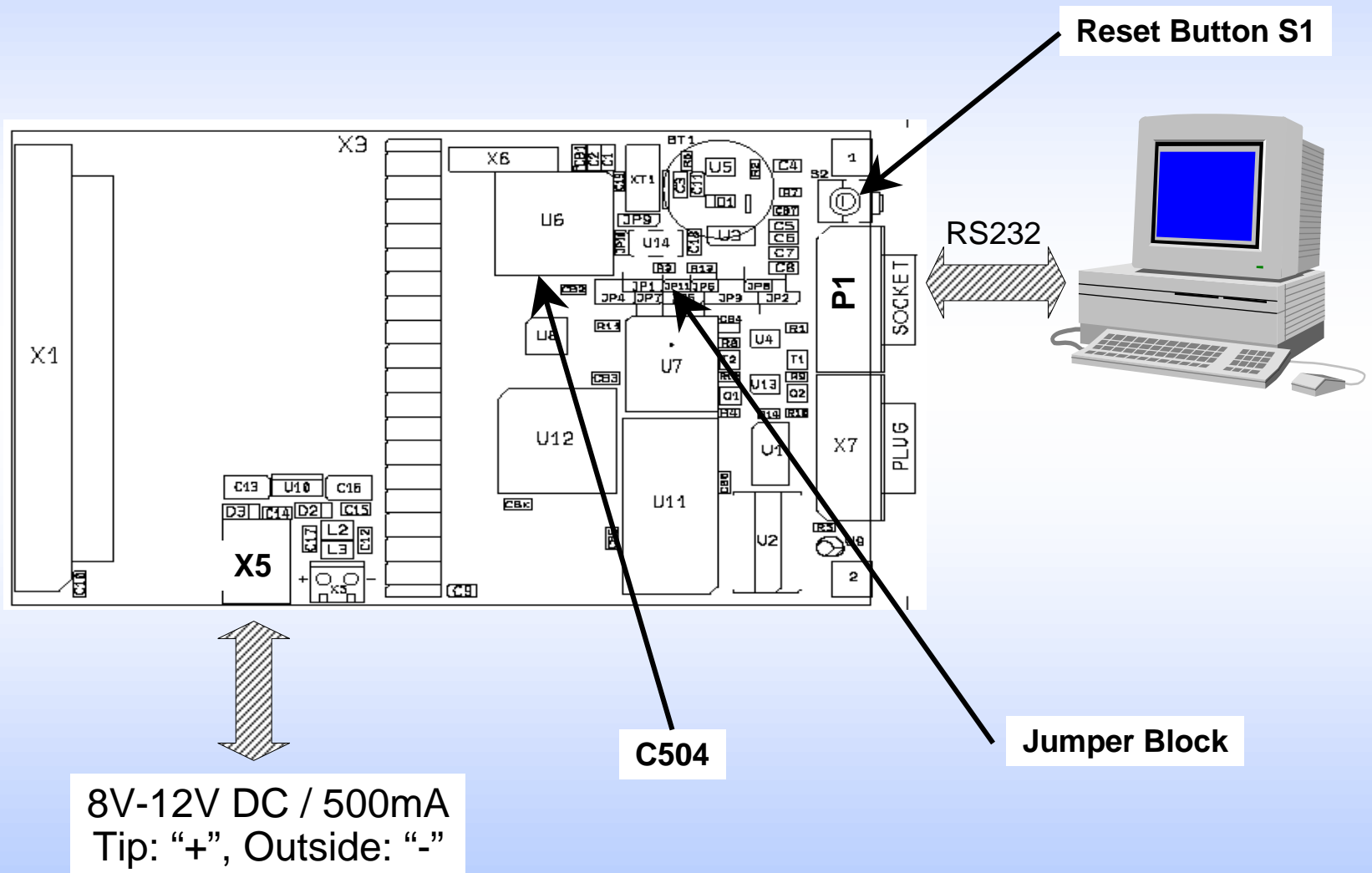
- Connect your Starter Kit board to your IBM compatible PC using a serial cable (connector X4 to COM1 - X4 is the connector closest to the RESET button on the board)**
  - Attention Windows 95 users:  
Disable the FIFO Buffer in Settings/Control Panel/System/Device Manager/Ports/COM1/Port Settings/Advanced!
- Make sure that all jumpers are in their default positions as described in the KitCON-504 Hardware-Manual.**
- Make sure that jumper JP11 is open.**
- Attach an unregulated power supply with 8V to 12V /500 mA to X2 on the KitCON-504. Double check the correct polarity.**
- Press the RESET button on the board.**

**NOTE:** These examples assume that the KitCON-504 is equipped with a 40 MHz crystal. Other editions of the board may have slower crystals. This will effect the timing of periodic interrupts, and some of the DAVe settings.

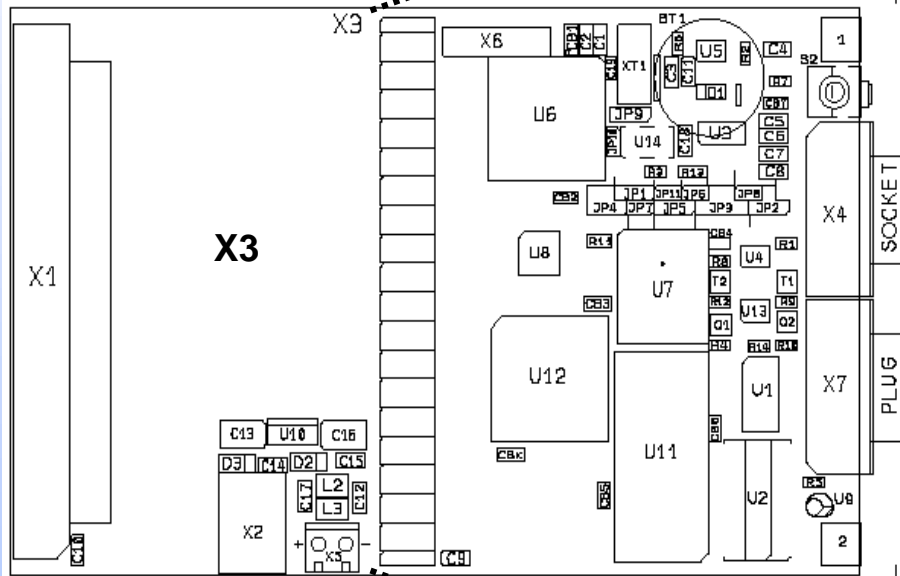
**For more information, read the Hardware Manual on the Starter Kit CD ROM**



# How to set up your system: KitCON-504 HW setup (cont.)



# How to set up your system: kitCON-504 X3 connector layout



You will be needing  
this page  
when you do  
the exercises...

Supply Voltage	VCC	VCC	GND	GND
Data-Bus	D0	D2	D4	D6
	D1	D3	D5	D7
Address-Bus	A0	A2	A4	A6
	A1	A3	A5	A7
	A8	A10	A12	A14
	A9	A11	A13	A15
Control-Signals	/RD	/PSEN	RES	/RES
	/WR	ALE	/EA	MDIS
	/CS0	/CS2		Reserved 2
	/CS1			Reserved 1
Special Purpose	Reserved 3	Reserved 4	/INT0	
	Reserved 3	Reserved 4		
Analog Input	P1.0	P1.2	P1.4	P1.6
	P1.1	P1.3	P1.5	P1.7
Digital-Port P1	P1.0	P1.2	P1.4	P1.6
	P1.1	P1.3	P1.5	P1.7
Digital-Port P3	P3.0, RxD	P3.2	P3.4	P3.6, /WR
	P3.1, TxD	P3.3	P3.5	P3.7, /RD
Parallel I/O (optional)	PA0	PA2	PA4	PA6
	PA1	PA3	PA5	PA7
	PB0	PB2	PB4	PB6
	PB1	PB3	PB5	PB7
	PC0	PC2	PC4	PC6
	PC1	PC3	PC5	PC7
Supply Voltage	VCC	VCC	GND	GND

# How to set up your system: DAvE 2.0 installation

---

- Insert the DAvE 2.0 CD in your CD ROM drive.**
- Run CD ROM\Setup.exe.**
- Follow the setup program's instructions.**
  - Install the microcontrollers you'd like to work with. You can always re-install or delete derivatives later on.
  - Be sure to install the Smart Search.
  - If you'd like to get a detailed introduction to DAvE 2.0, be sure to include the tutorial. You can delete it afterwards to free up space.
- If you don't have Acrobat Reader installed on your PC, do so by choosing to install Acrobat Reader which is included on the DAvE CD.**  
**(Hint: You need Acrobat Reader 4.0 or higher)**
- Please note:**  
**You need WINDOWS 95 or later or WINDOWS NT in order to run DAvE!**

# How to set up your system: Keil uVision installation

---

- Insert the Starter Kit CD into your CD ROM drive.**
  
- Start welcome.pdf from the CD ROM  
(Acrobat Reader is required but also included on the CD:  
CD ROM\install\reader\)**
  
- Go to Third Party Development Tools**
  
- Go to Compiler/Assembler**
  
- Go to Keil Elektronik**
  
- Choose EK51 installation. The following text will assume that Keil  
tool chain is installed in the default directory c:\c51eval\**

# How to set up your system: Exercise directory structure

---

- ❑ **Create a subdirectory on your hard drive:**

c:\hotC504\_1

- ❑ **Create the following subdirectories on you hard drive to store the project and code for each of the exercises:**

c:\hot504\_1\4URT\_1

c:\hot504\_1\4URT\_2

c:\hot504\_1\4ADC\_1

c:\hot504\_1\4CCU\_1

c:\hot504\_1\4CCU\_2

# Files to build a Program

---

- ❑ **Source files containing executable C statements**
  - Extension: .C (created by DAVe)
  
- ❑ **Header files containing function prototypes and definitions**
  - Extension: .H (created by DAVe)
  
- ❑ **Project File containing all microcontroller configurations:**
  - Extension: .DAV (created by DAVe)
  
- ❑ **A  $\mu$ Vision project control file containing the list of source files in the project and the compiler controls etc.:**
  - Extension: .PRJ

# C51 emitted file types

---

## ❑ **Projectname . OBJ**

- Unlocated object file with no absolute addresses assigned.

## ❑ **Projectname . LST**

- A list file containing the original source lines but with any errors or warnings indicated

## ❑ **Projectname . SRC**

- An optional file containing the assembler code generated by the compilation process.
- This can be assembled with A51 to produce the .OBJ file as an alternative to going straight to a .OBJ file from the compiler.

## ❑ **Projectname . ERR**

- A summary of the errors and warnings that occurred during compilation

# L51 linker emitted files

---

## ❑ **Projectname .<noextension>**

- Absolute object file in the extended OMF-51 format, containing a binary representation of the program plus optionally, debug symbol information. It also holds the necessary make information to allow  $\mu$ Vision to rebuild the program.

## ❑ **Projectname . M51**

- A map file containing the address and length of all classes, sections and register banks.

## ❑ **Projectname . REG**

- A special file which holds information on all the register masks generated by every function in the program. By giving this file back to the compiler, it can be used to allow global register usage optimization.

## ❑ **Projectname . LNK**

- A linker control file automatically generated by  $\mu$ Vision2 which contains the user's own .LIN linker control file (if used) with the object file list prepended to it.

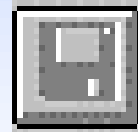


# uVision's Toolbar

---

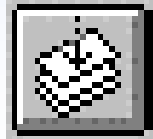


**Edit File**

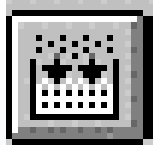


**Save File**

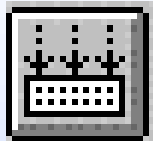
---



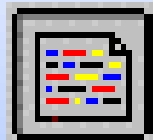
**Compile File** - Compile file being edit ed



**Update Project** - Compile any file that has been edited since last build



**Build Project** - Compile all files and link




**Enable Editor's Syntax Colouring**


# Hints regarding DAVe 2.0

---

## ❑ To create a new Project with DAVe:

- Select “File | New” from the pull down menu or press 
- Select your microcontroller and click “create”



## ❑ To generate code with DAVe:

- Select “File | Generate Code” from the pull down menu or press 

## ❑ To configure a peripheral:

- Move your mouse over the peripheral when DAVe shows the block diagram and click the left mouse button

## ❑ To get context specific help in DAVe:

- Move your mouse over the item you want to find out about, sometimes a yellow text window will pop up
- When configuring a peripheral, click the info-button arrow  to go to the corresponding User’s Manual chapter
- Select “View | Register Explorer” or press 

## ❑ Validate each alpha numeric entry by pressing ENTER

## ❑ To save & close any DAVe window, click or

# Hints regarding the Assembler Startup File

---

- ❑ **The Keil Assembler Startup File ('startup.a51') is meant to be executed after reset to provide the startup sequence for the microcontroller:**
  - Basic microcontroller initialization (e.g. enabling the XRAM)
  - Initialization of static variables in internal or external RAM
  - Call the user program: main().
- ❑ **DAvE 2.0 no longer incorporates the Assembler Startup Files of the Keil and the Tasking Compilers (like DAvE 1.0 did)**
- ❑ **Instead, DAvE 2.0 generates two new files:**
  - a file called ***projectname.dpt*** which in the future will be used by your programming environment to retrieve information about the generated files or the actual project values.
  - A file called ***projectname.asm*** which contains the assembler formatted information about the actual project values. It will in the future be used by your programming environment.

# Hints regarding the Assembler Startup File (cont.)

---

❑ **However:**

**When using the examples in this training material, the XRAM is not used and therefore it is not required to include an assembler startup file (the default startup file will be included automatically by the compiler).**

# Hints regarding the Exercises

---

## ❑ Exercise numbering scheme:

- Generally the exercise name contains the starter kit, the peripheral the exercise was mainly designed for and a running number
- Exercise “4URT\_2” would be the 2nd example for the USART using the kitCON-504 Starter Kit

## ❑ To create a new exercise:

- For each exercise, use a different subdirectory c:\hot504\_1\newex (e.g. c:\hot504\_1\4URT\_1)

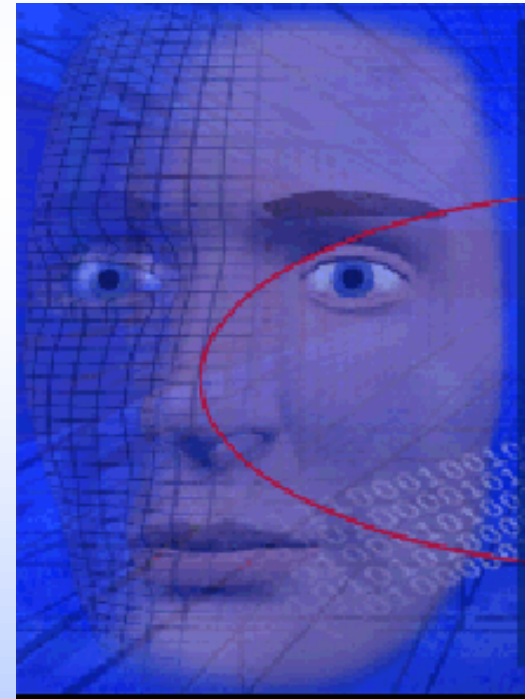
# Hints regarding the Exercises (cont.)

---

- ❑ **The exercises are created in a way that they use only peripherals which have been introduced already (either directly before the exercise or in previous exercises)**
- ❑ **If an exercise does not work when running the debugger:**
  - Check the oscilloscope connections (right pin?)
  - Check the DAVe /  $\mu$ Vision configurations
  - Re-generate the code with DAVe (USER CODE remains!)
  - Recompile the exercise in  $\mu$ Vision
  - Reload the exercise into the debugger and run it again
  - dScope may appear to react slowly when using Windows NT

---

Let's get  
started now!



# Universal Sync./Async. Receiver Transmitter (USART)

---

## ❑ Baud Rate Generation

- External Oscillator
- Timer 2
- Timer 1
- Timer 2 and Timer 1 (one for receive and one for transmit)

## ❑ Mode 0

- Synchronous Mode (Half Duplex)
- Fixed Baud Rate ( $f_{osc}/12$ )
- 8 data bits (LSB first)
- TxD Pin used as shift clock
- RxD Pin used for data

## ❑ Mode 1

- Asynchronous Mode (Full Duplex)
- Variable Baud Rate
- 8 data bits (LSB first)
- 1 Start Bit (0); 1 Stop Bit (1)



# Universal Sync./Async. Receiver Transmitter (USART) - cont.

---

## ❑ Mode 2

- Asynchronous Mode (Full Duplex)
- Fixed Baud Rate ( $f_{osc}/32$  or  $f_{osc}/64$ )
- 9 data bits (LSB first) -- Parity, Multiprocessor Communication
- 1 Start Bit (0); 1 Stop Bit (1)

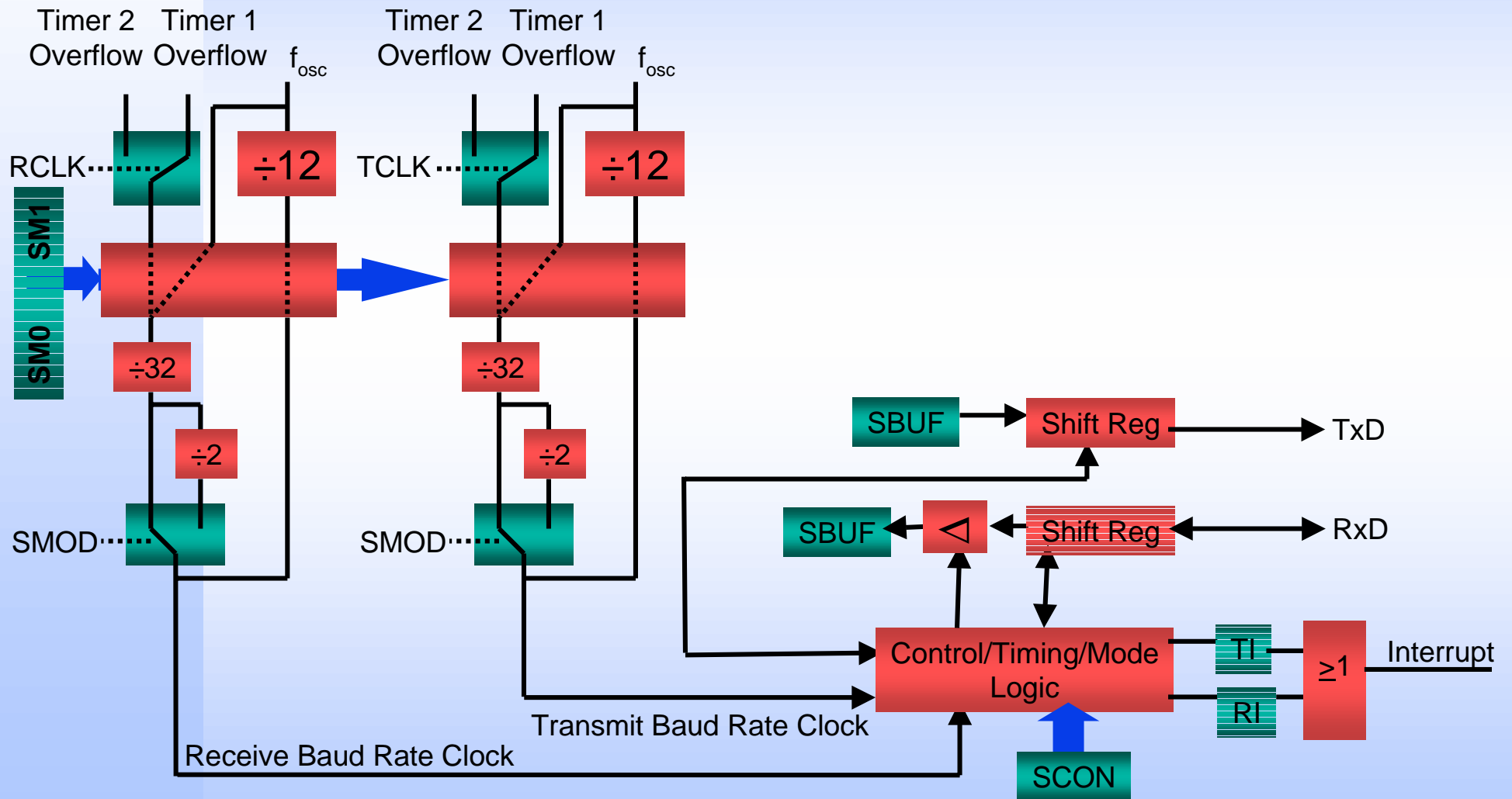
## ❑ Mode 3

- Asynchronous Mode (Full Duplex)
- Variable Baud Rate
- 9 data bits (LSB first) -- Parity, Multiprocessor Communication
- 1 Start Bit (0); 1 Stop Bit (1)

## ❑ Interrupts

- After data reception
- After data transmission
- After data reception & 9th data bit is a "1"
- After data reception & no errors

# USART Function Diagram



# Timer/Counter 0 and 1

---

## ❑ 2 separate 16 Bit Timers/Counters

- Timer when clocked by external oscillator ( $f_{osc} \div 6$ )
- Counter when clocked by external pin (P3.4/T0 or P3.5/T1)
- Can measure pulse widths when used as a “Gated Timer”
  - Clocked by external oscillator when P3.2/INT0 or P3.3/INT1 is high

## ❑ 4 Operating Modes

- Mode 0
  - 8 bit timer/counter with 5 bit (divide by 32) prescaler
  - Interrupt on rollover (if desired)
- Mode 1
  - 16 bit timer/counter
  - Interrupt on rollover (if desired)
- Mode 2
  - 8 bit timer with 8 bit reload on overflow
  - Interrupt on rollover (if desired)

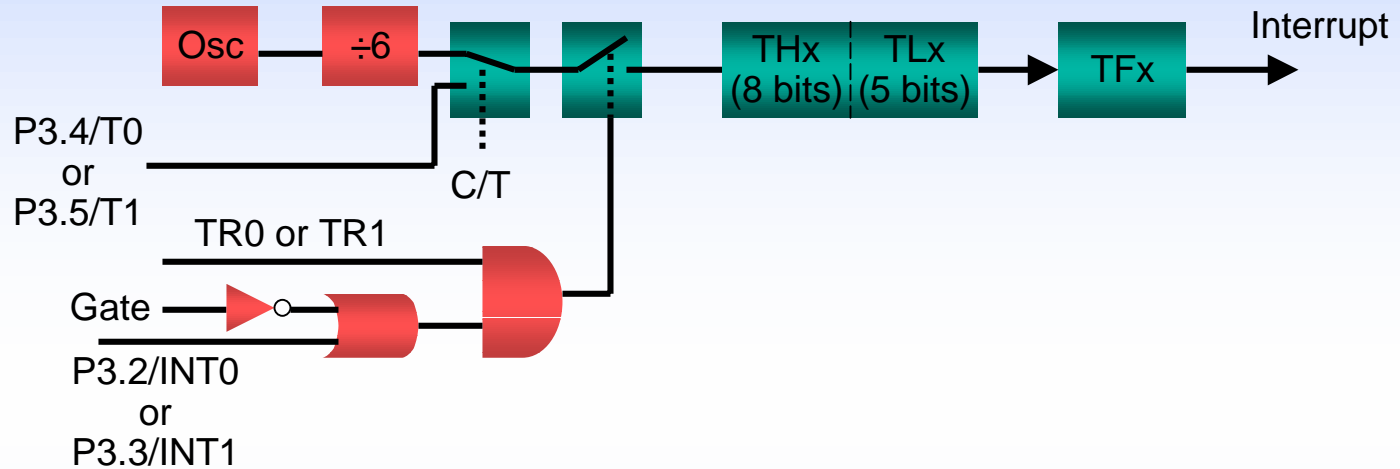
# Timer/Counter 0 and 1 (cont.)

---

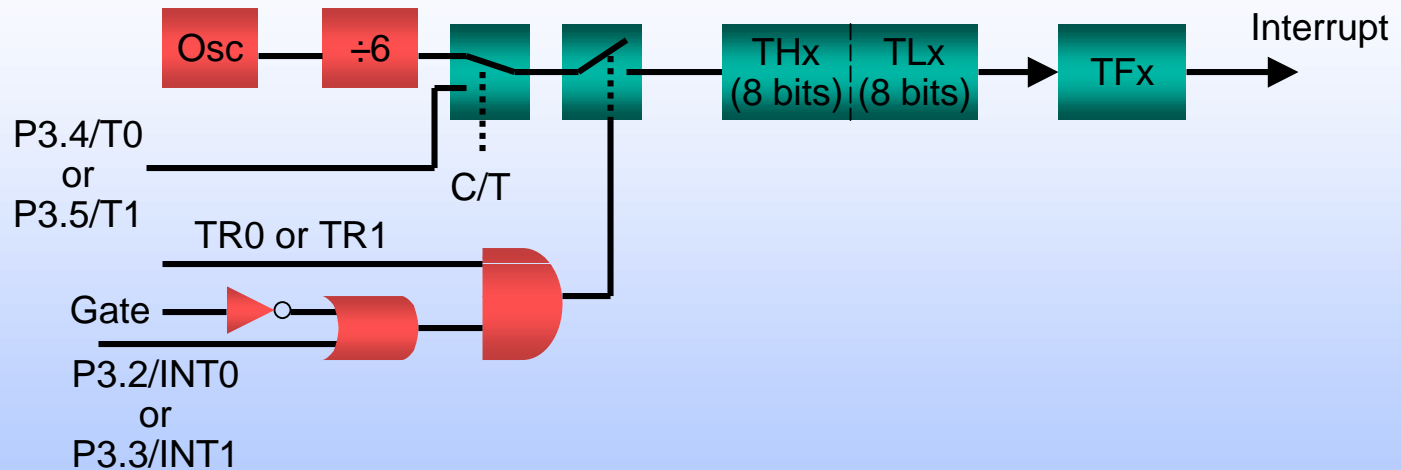
- Mode 3
  - Timer 0 becomes 2 separate 8 bit timers/counters with individual interrupts on overflow (if desired)
  - Timer 1 becomes a 16 bit timer that can be started and stopped at any time, but cannot cause an interrupt
  - Useful when an “extra” timer is needed

# Timer/Counter 0 and 1 Operating Modes

## MODE 0



## MODE 1





# Exercise 4URT\_1 - Periodic Asynchronous Serial Data Transmission - Description



---

- ❑ **Objective: Send asynchronous serial data periodically using the USART, Timer 1 and Timer 0**
  - Period: 1 overflow of Timer 0 in mode 0 (2.458 ms with a 40 MHz crystal)
  - Data: Send constant data (0x55) at a baud rate of 9600
  - Use USART mode 1 (8 data bits, 1 start bit, 1 stop bit, variable baud rate)
  - Use Timer 1 to generate baudrate clock

# Exercise 4URT\_1 - DAVe Configurations

- ❑ **Start DAVe 2.0**



- ❑ **Select “Create a new project” from the Startup Dialog or click** 
- ❑ **Select the 8-Bit microcontroller C504 and click “Create”**  
(if this microcontroller is not on the list, you need to re-install it from the DAVe 2.0 CD ROM)
- ❑ **DAVe will create the project**
- ❑ **Save your project by selecting “File | Save” or press** 
  - Browse to directory “c:\hot504\_1\4URT\_1\”
  - Enter project name: “4URT\_1”
  - Click “Save”
- ❑ **You will see the C504 block diagram and the Project Settings Window (configuration see next slide)**
- ❑ **To get back to the Project Settings window in case you close it: Select “File | Project Settings”**



# Exercise 4URT\_1 - DAvE Configurations (cont.)

---

## ❑ Project Settings:

- General:
  - Controller Type:
    - C504-L40
- System Clock:
  - External Oscillator Frequency: Set to 40 MHz (or the crystal frequency of your KitCON-504 board)
- Close

# Exercise 4URT\_1 - DAvE Configurations (cont.)

---

## ❑ **Configure T0 (Timer/Counter 0):**

- Timer 0
  - Timer Mode (M1, M0):
    - 8 bit timer/counter (TH0) with divide-by-32 prescaler (TL0)
  - Timer Options:
    - Run Timer (TR0)
  - Interrupt Control:
    - Enable Timer 0 interrupt (ET0)
- Functions
  - T01v\_Init
- Close

# Exercise 4URT\_1 - DAvE Configurations (cont.)

---

## ❑ **Configure 8-Bit USART**

- Control:
  - Global:
    - Use TXD (P3.1) for serial Channel
  - Mode Control (SM0, SM1)
    - Mode 1: 8-bit data, 1 startbit, 1 stopbit, variable baudrate (async)
- Functions
  - USART\_vInit
  - USART\_vSendData
- Close

# Exercise 4URT\_1 - DAVe Configurations (cont.)

---

## **Configure T1 (Timer/Counter 1) for baudrate generation:**

- Timer 1
  - Timer Mode (M1, M0):
    - 8 bit timer/counter (TL1) with 8-bit auto-reload (TH1)
  - Timer Options:
    - Run Timer (TR1)
  - Timer Register:
    - Timer register High (TH1) = 0xF5 (or whatever number get the Baudrate closest to 9600)
- Functions
  - T01v\_Init
- Close

## **Generate Code ( )**

## **DAVe will show you all the files that he has generated (File Viewer is opened automatically)**

# Exercise 4URT\_1 - uVision Configurations

---

- ❑ **Create new Project: c:\hot504\_1\4URT\_1\4URT\_1.prj**
  
- ❑ **Edit Project:**
  - Add MAIN.C (from c:\hot504\_1\4URT\_1)
  - Add T01.C (from c:\hot504\_1\4URT\_1)
  - Add USART.C (from c:\hot504\_1\4URT\_1)
  - Open All
  - Save
  
- ❑ **Options/Environment Pathspecs...**
  - In most cases the “Automatically determine path specifications” will work, however if you have problems building the project you may have to enter the paths in manually.

# Exercise 4URT\_1 - uVision Configurations (cont.)

---

## ❑ Edit MAIN.C:

- include endless loop in main():

```
// USER CODE BEGIN (Main,3)
while(1) {};
// USER CODE END
```

## ❑ Edit T01.C:

- transmit serial data when the Timer 0 ISR (T01\_vilSrTmr0) is executed:

```
// USER CODE BEGIN (T01_IsrTmr0,1)
USART_vSendData(0x55);
// USER CODE END
```

## ❑ Build All

- This will compile and link the project and create an object file and hex file.

# Exercise 4URT\_1 - Running the Program

---

## □ In general there are three ways to run the program

- Using the Phytec Flash-Tools and the FLASHT.exe terminal program for DOS
  - The program (hex file) can be programmed into the KitCON non-volatile flash memory. The board can then operate as a stand-alone device (no need for the PC).
    - This will overwrite any software that was previously located in the flash memory (including the Keil monitor).
- Using the Keil monitor (located in the KitCON flash memory) and dScope for Windows
  - The program (object file) can be downloaded into the KitCON RAM and executed from Windows
- Using the Keil monitor (located in the KitCON flash memory) and the MON51.exe terminal program for DOS
  - The program (hex file) can be downloaded into the KitCON RAM and executed from DOS

# Exercise 4URT\_1 - Running the Program (cont.)

---

- ❑ **For all of the examples we will use the Keil monitor and dScope to download and execute programs.**
  - To use the Keil tools, the monitor firmware must be programmed into the KitCON-504 flash memory.
    - The monitor comes pre-programmed on the flash, but downloading any other code into the flash will erase the monitor. In this case the flash will have to be reprogrammed.
    - The monitor firmware is located on the starter kit CD-ROM in the `\cdrom\startkit\c504\monitor\keil` directory
    - The `Flasht.exe` programming tool is located in the `\cdrom\startkit\c504\flash` directory
  - For instructions on programming the flash, see the KitCON-504 Hardware Manual



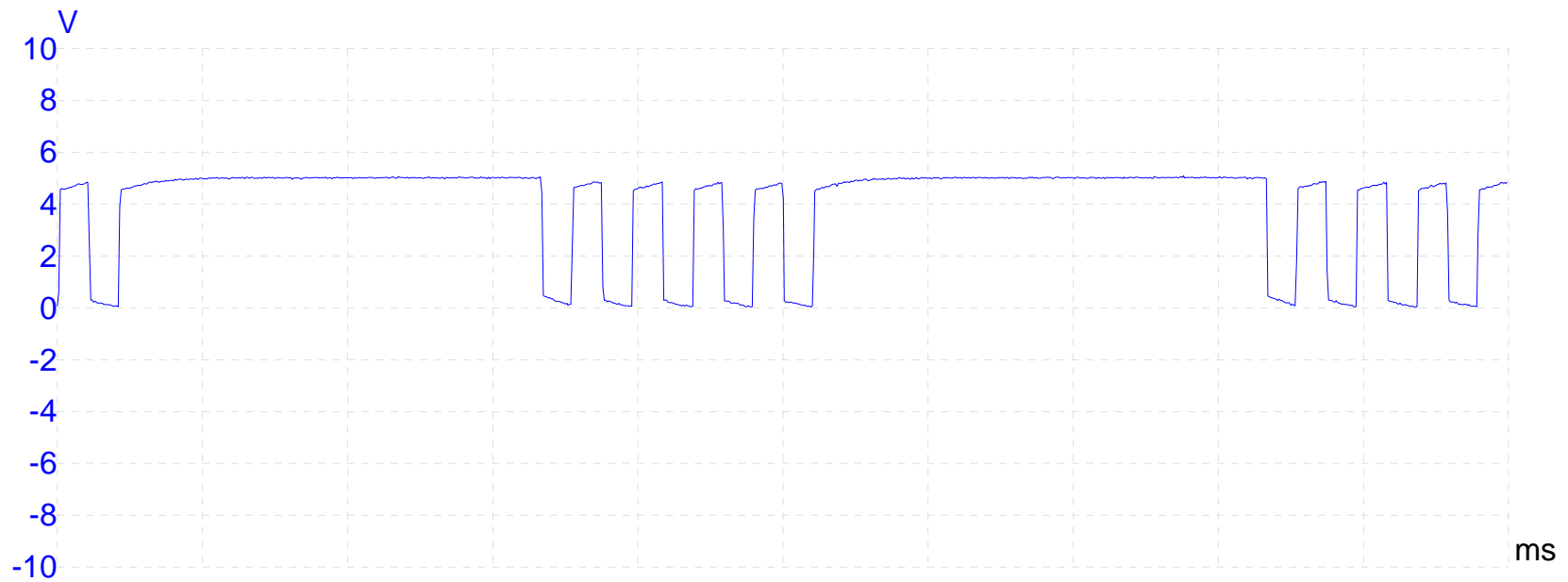
# Exercise 4URT\_1 - Running the Program (cont.)

---

- **Run dScope Debugger**
- **File | Load CPU Driver: mon51.dll**  
this will establish a connection between the Keil Monitor (located in the flash) and dScope (once this happens dScope becomes tScope - the target debugger)
- **File | Load Object file: Load c:\hot504\_1\4URT\_1\4URT\_1**
- **Connect Oscilloscope to P3.1 / TxD (connector X3 pin 97)**
- **Hit Go!**
- **When you are finished viewing the output, push the RESET button on the KitCON-504**

# Exercise 4URT\_1 - Screenshot

---



P3.1/TxD  
Example 4URT\_1

0.5 ms / div

# Exercise 4URT\_2 - Periodic Synchronous Serial Data Transmission - Description



---

- ❑ **Objective: Send synchronous serial data periodically using the USART and Timer 1**
  - Period: 16 Timer 1 counts
    - Timer 1 in mode 2
      - Reload value of 0xF0 overflow of Timer 1 in mode 0 (4.8  $\mu$ s with a 40 MHz crystal)
  - Data: Send constant data (0xAA)
  - Use USART mode 0
    - 8 data bits
    - Fixed Baud Rate
      - 1/12 external oscillator frequency
      - 3.333 Mb/s (@ 40 MHz)

# Exercise 4URT\_2 - DAVe Configurations

- ❑ Start DAVe 2.0



- ❑ Select “Create a new project” from the Startup Dialog or click 
- ❑ Select the 8-Bit microcontroller C504 and click “Create” (if this microcontroller is not on the list, you need to re-install it from the DAVe 2.0 CD ROM)
- ❑ DAVe will create the project
- ❑ Save your project by selecting “File | Save” or press 
  - Browse to directory “c:\hot504\_1\4URT\_2\”
  - Enter project name: “4URT\_2”
  - Click “Save”
- ❑ You will see the C504 block diagram and the Project Settings Window (configuration see next slide)
- ❑ To get back to the Project Settings window in case you close it: Select “File | Project Settings”

# Exercise 4URT\_2 - DAVe Configurations

---

## □ Project Settings:

- General:
  - Controller Type:
    - C504-L40
- System Clock:
  - External Oscillator Frequency: Set to 40 MHz (or the crystal frequency of your KitCON-504 board)
- Close

# Exercise 4URT\_2 - DAVe Configurations (cont.)

---

## ❑ **Configure T1 (Timer/Counter 1):**

- Timer 1
  - Timer Mode (M1, M0):
    - 8 bit timer/counter (TL1) with 8-bit auto-reload (TH1)
  - Timer Register
    - Timer auto-reload = 0xF0
  - Timer Options:
    - Run Timer (TR1)
  - Interrupt Control:
    - Enable Timer 1 interrupt (ET1)
- Functions
  - T01v\_Init
- Close

# Exercise 4URT\_2 - DAVe Configurations (cont.)

---

## **Configure 8-Bit USART**

- Control:
  - Global:
    - Use TXD (P3.1) for serial Channel
  - Mode Control (SM0, SM1)
    - Mode 0: 8-bit Shift register, fixed baud rate (sync)
- Functions
  - USART\_vInit
  - USART\_vSendData
- Close

## **Generate Code ( )**

## **DAvE will show you all the files that he has generated (File Viewer is opened automatically)**

# Exercise 4URT\_2 - uVision Configurations

---

- ❑ **Create new Project: c:\hot504\_1\4URT\_2\4URT\_2.prj**
  
- ❑ **Edit Project:**
  - Add MAIN.C (from c:\hot504\_1\4URT\_2)
  - Add T01.C (from c:\hot504\_1\4URT\_2)
  - Add USART.C (from c:\hot504\_1\4URT\_2)
  - Open All
  - Save
  
- ❑ **Options/Environment Pathspecs...**
  - In most cases the “Automatically determine path specifications” will work, however if you have problems building the project you may have to enter the paths in manually.



# Exercise 4URT\_2 - uVision Configurations (cont.)

---

## ❑ Edit MAIN.C:

- include endless loop in main():

```
// USER CODE BEGIN (Main,3)
while(1) {};
// USER CODE END
```

## ❑ Edit T01.C:

- transmit serial data when the Timer 1 ISR (T01\_vilSrTmr1) is executed:

```
// USER CODE BEGIN (T01_IsrTmr1,1)
USART_vSendData(0xAA);
// USER CODE END
```

## ❑ Build All

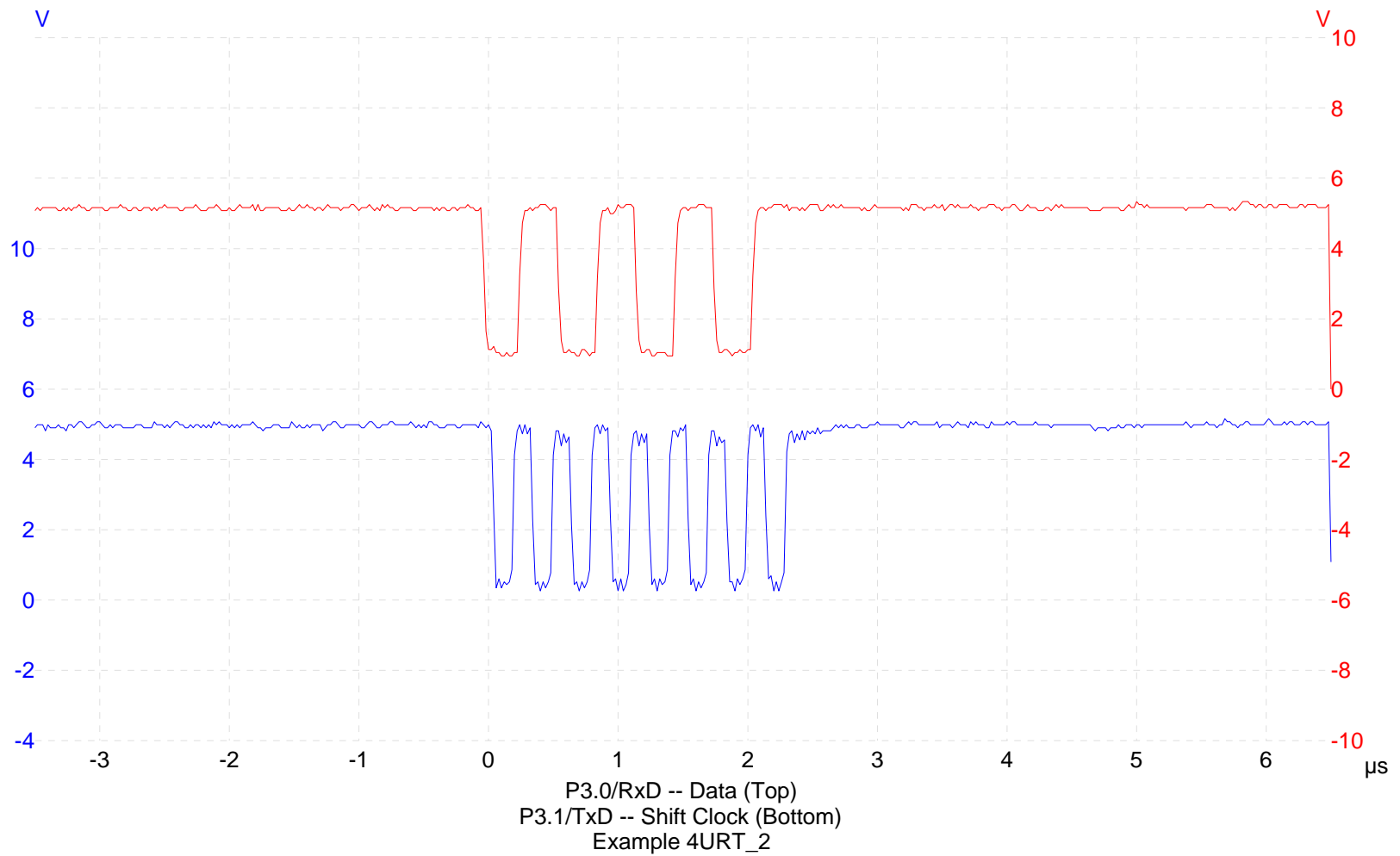
- This will compile and link the project and create an object file and hex file.

# Exercise 4URT\_2 - Running the Program

---

- **Run dScope Debugger**
  
- **File | Load CPU Driver: mon51.dll**  
this will establish a connection between the Keil Monitor (located in the flash) and dScope (once this happens dScope becomes tScope - the target debugger)
  
- **File | Load Object file: Load c:\hot504\_1\4URT\_2\4URT\_2**
  
- **Connect Oscilloscope to P3.1 / TxD (connector X3 pin 97) and to P3.0 / RxD (connector X3 pin 93)**
  
- **Hit Go!**
  
- **When you are finished viewing the output, push the RESET button on the KitCON-504**

# Exercise 4URT\_2 - Screenshot

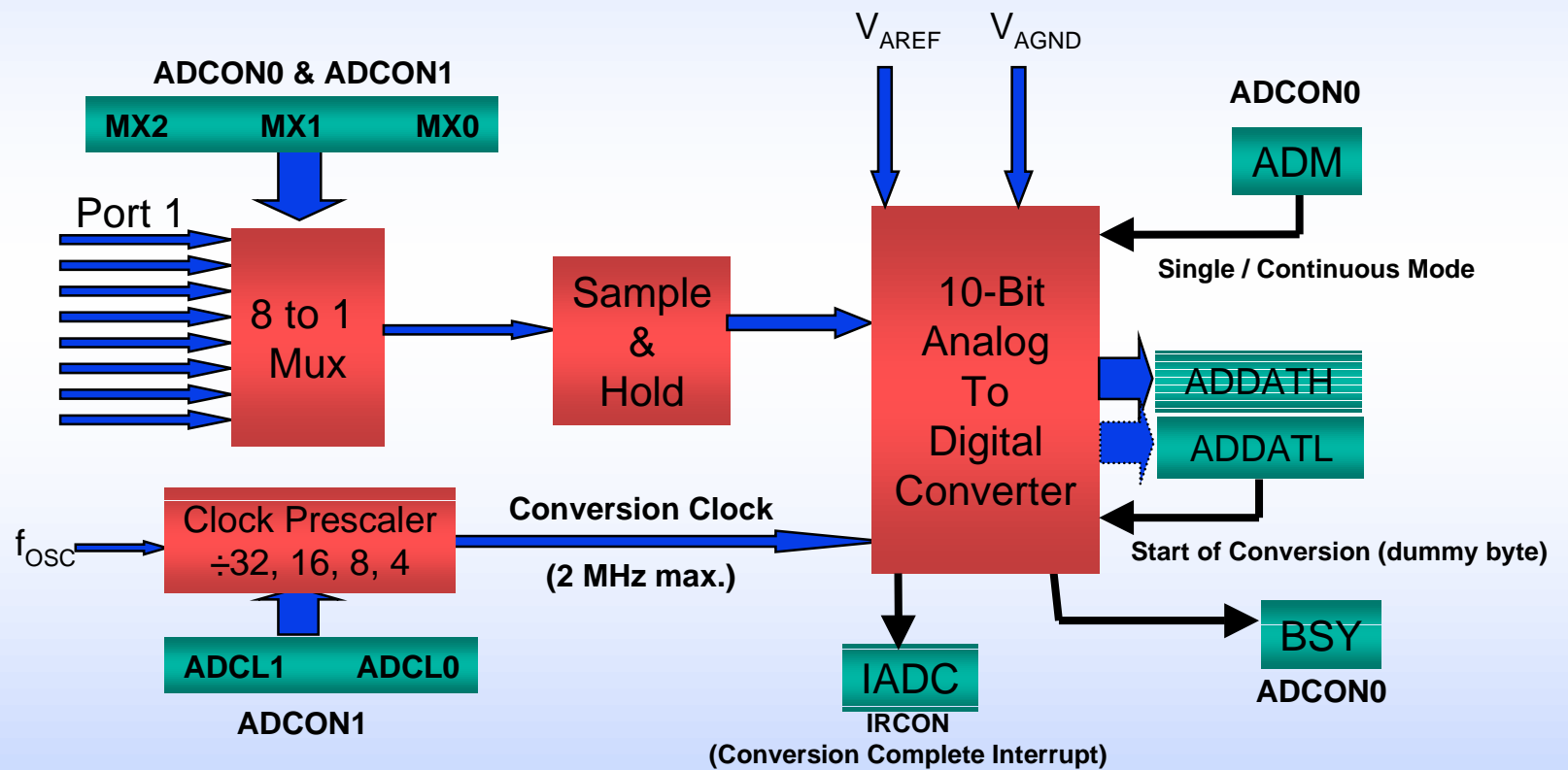


# Analog-to-Digital Converter

---

- 8 channel multiplexed input**
- Programmable conversion time**
- 10 bit resolution**
- 6  $\mu$ sec minimum conversion time**
- Internal Capture and Hold circuit**
- Single or Continuous conversion modes**
- Busy Flag**
- End of Conversion interrupt**

# Analog-to-Digital Converter - Functional Diagram



# Timer/Counter 2

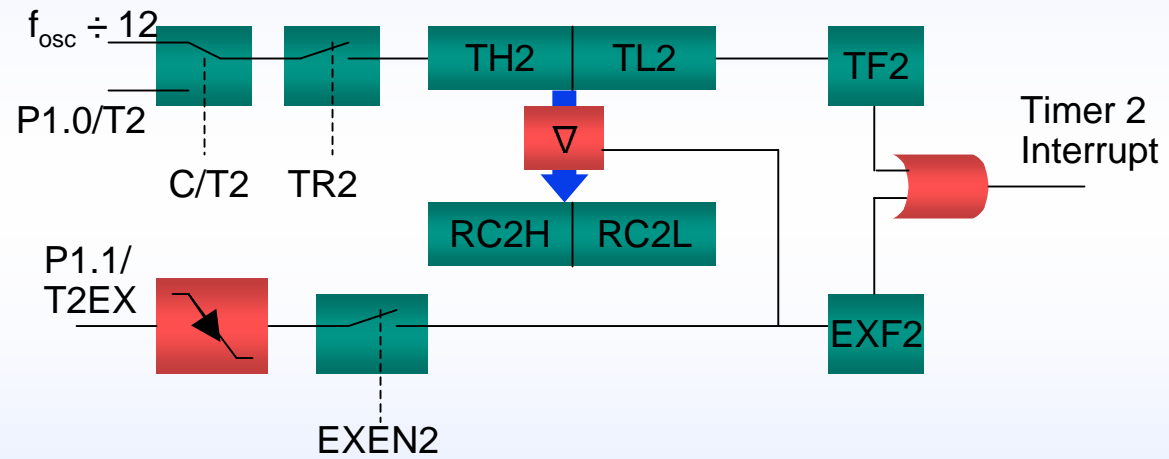
---

- ❑ **16-bit Timer**
  - Counts at 1/12 external oscillator frequency (instruction cycle time)
- ❑ **16-bit Counter**
  - Counts high-to-low transitions on pin P1.0/T2
- ❑ **16-bit Capture**
  - Timer value captured on high-to-low transition of pin P1.1/T2EX
- ❑ **16-bit Reload**
  - On overflow if up counting
  - On underflow if down counting
  - In response to high-to-low transition of pin P1.1/T2EX
- ❑ **Baudrate generation for USART**
- ❑ **Up or Down Counting**
  - Up Counting if P1.1/T2EX = 1, Down Counting if P1.1/T2EX = 0
- ❑ **Interrupt Generation**



# Timer 2

## Capture Mode





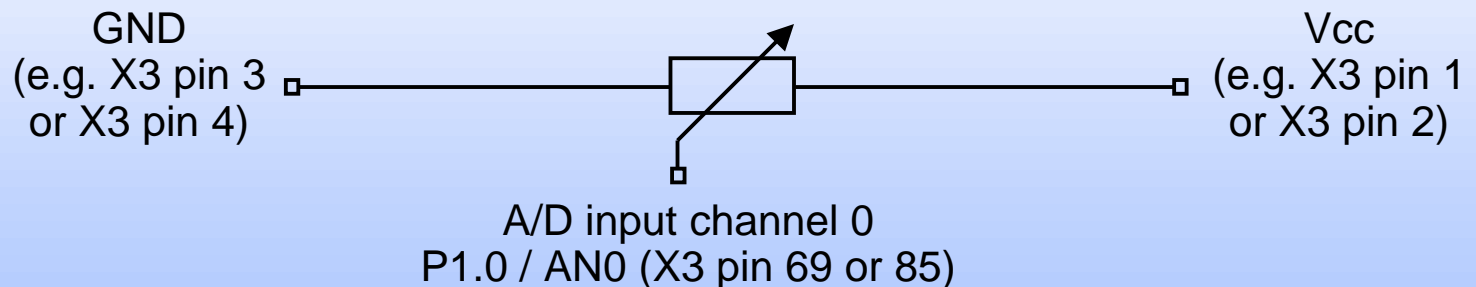
# Exercise 4ADC\_1 - Generate a variable frequency square wave - Description

## ❑ Objective:

- Control the frequency of a square wave using Timer 2 and an ADC Value from the ADC channel 0.
  - ADC will be placed in “Continuous Conversion Mode”
  - ADC ISR will update the square wave frequency
  - Timer 2 will be placed in Up Counting Mode
  - Timer 2 ISR will toggle pin P1.2

## ❑ KitCON-504 Configuration:



- Connect  $V_{AREF}$  and  $V_{AGND}$  to  $V_{CC}$  and GND (respectively)
  - connect X3 pin 1 to X3 pin 61
  - connect X3 pin 3 to X3 pin 62
- Connect a potentiometer to A/D channel 0 (pin P1.0 - connector X3 pin 69 or 85)



# Exercise 4ADC\_1 - DAVe Configurations

- ❑ **Start DAVe 2.0**



- ❑ **Select “Create a new project” from the Startup Dialog or click** 
- ❑ **Select the 8-Bit microcontroller C504 and click “Create” (if this microcontroller is not on the list, you need to re-install it from the DAVe 2.0 CD ROM)**
- ❑ **DAVe will create the project**
- ❑ **Save your project by selecting “File | Save” or press** 
  - Browse to directory “c:\hot504\_1\4ADC\_1\”
  - Enter project name: “4ADC\_1”
  - Click “Save”
- ❑ **You will see the C504 block diagram and the Project Settings Window (configuration see next slide)**
- ❑ **To get back to the Project Settings window in case you close it: Select “File | Project Settings”**

# Exercise 4ADC\_1 - DAvE Configurations (cont.)

---

## ❑ Project Settings:

- General:
  - Controller Type:
    - C504-L40
- System Clock:
  - External Oscillator Frequency: Set to 40 MHz (or the crystal frequency of your KitCON-504 board)
- Close

# Exercise 4ADC\_1 - DAvE Configurations (cont.)

---

## ❑ **Configure Timer 2:**

- Timer 2:
  - Start / Stop Control:
    - Start timer 2 after initialization (TR2)
  - Interrupt Control:
    - Enable Timer 2 interrupt (ET2)
  
- Functions:
  - T2\_vInit
  - T2\_uwSetRC2Reg
  
- Close

# Exercise 4ADC\_1 - DAvE Configurations (cont.)

---

## ❑ **Configure Port 1:**

- Port 0/1
  - Port 1
    - Use P1.2 / AN2 / CC0 as general IO
- Functions
  - Generic Port Function Library
    - IO\_vTogglePin
- Close

# Exercise 4ADC\_1 - DAvE Configurations (cont.)

---

## **Configure 10-bit ADC:**

- Control:
  - Input Selection:
    - Use P1.0 for ADC channel 0
  - Conversion Options:
    - Continuous conversion (ADM)
  - Interrupt Control:
    - Enable A/D interrupt (EADC)
  
- Functions:
  - ADC\_vInit
  - ADC\_vStart
  - ADC\_uwRead10Conv
  
- Save & Close

## **Generate Code ( )**

## **DAvE will show you all the files that he has generated (File Viewer is opened automatically)**

# Exercise 4ADC\_1 - uVision Configurations

---

- Create new Project: c:\hot504\_1\4ADC\_1\4ADC\_1.prj**
  
- Edit Project:**
  - Add MAIN.C (from c:\hot504\_1\4ADC\_1)
  - Add T2.C (from c:\hot504\_1\4ADC\_1)
  - Add IO.C (from c:\hot504\_1\4ADC\_1)
  - Add ADC.C (from c:\hot504\_1\4ADC\_1)
  - Open All
  - Save
  
- Options/Environment Pathspecs...**
  - In most cases the “Automatically determine path specifications” will work, however if you have problems building the project you may have to enter the paths in manually.

# Exercise 4ADC\_1 - uVision Configurations (cont.)

---

## ❑ Edit MAIN.C:

- start A/D converter and include endless loop in main():

```
// USER CODE BEGIN (Main,3)
ADC_vStart(CHANNEL_0, ADC_MODE_CONT);
while(1) {};
// USER CODE END
```

## ❑ Edit ADC.C:

- Move A/D result to the Timer 2 reload registers

```
// USER CODE BEGIN (ADC_Isr,0)
T2_vSetRC2Reg(ADC_uwRead10BitConv()<<6);
// USER CODE END
```



# Exercise 4ADC\_1 - uVision Configurations (cont.)

---

## ❑ Edit T2.C:

- Toggle P1.2 in T2\_vIsrTmr():

```
// USER CODE BEGIN (T2_IsrTmr,1)
IO_vTogglePin(P1_2);
// USER CODE END
```

## ❑ Build All

- This will compile and link the project and create an object file and hex file.

# Exercise 4ADC\_1 - Running the Program

---

- **Run dScope Debugger**
- **File | Load CPU Driver: mon51.dll**  
this will establish a connection between the Keil Monitor (located in the flash) and dScope (once this happens dScope becomes tScope - the target debugger)
- **File | Load Object file: Load c:\hot504\_1\4ADC\_1\4ADC\_1**
- **Connect Oscilloscope to P1.2 (connector X3 pin 86) and connect the potentiometer to P1.0 (connector X3 pin 69 or 85)**
- **Hit Go! And move the potentiometer.**
- **When you are finished viewing the output, push the RESET button on the KitCON-504**

# Capture / Compare Unit

---

## ❑ For General Purpose Use and Motor Control Applications

### ❑ 2 Timer Units

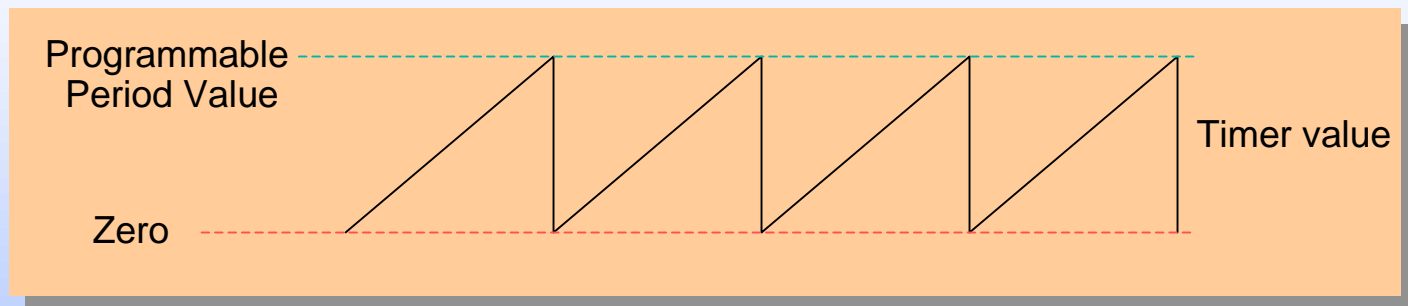
- Programmable Resolution
  - $f_{osc} \div 256$  to  $f_{osc} \div 2$  (6 times faster than instruction cycle)
- Up and Down Counting (CT1 only)
- Supports Center (CT1 only) and Edge aligned PWM generation
- 16 bit Timer
  - 3 channel Capture
  - 6 channel Compare (3 independent channels)
- 10 bit Timer
  - 1 Compare channel

### ❑ Motor Control Modes

- 3 phase Inverter control
- Brushless DC Motor Mode (Block Commutation)
- 4, 5, and 6 phase Motor Control
- Emergency Trap for protection

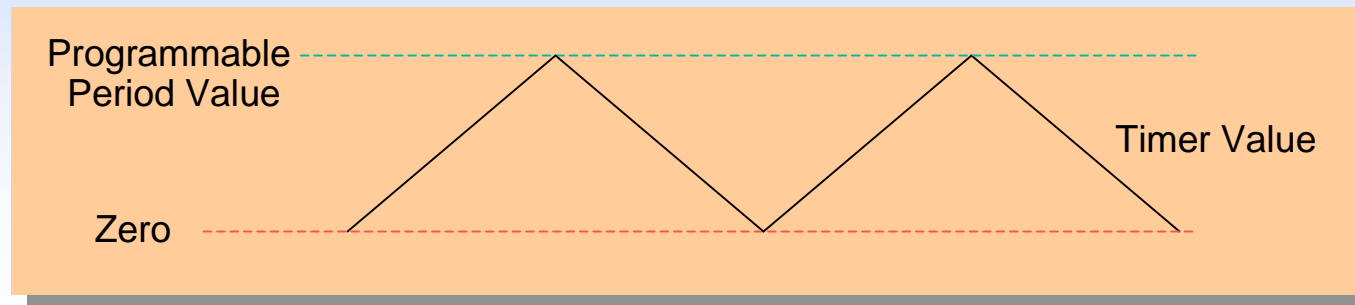
# Capture / Compare Unit Timers

- ❑ **Both the 16 and 10-bit timers count until they reach a programmable period value**
  - Period value (and compare values -- if used) can be written to a shadow register at any time.
  - Shadow registers are transferred to the CCU when the timer value reaches 0.
  
- ❑ **Timers can count in 2 different Modes**
  - Mode 0 (Edge Aligned):



# Capture / Compare Unit Timers (cont.)

- Mode 1 (Center Aligned -- CT1 Only):



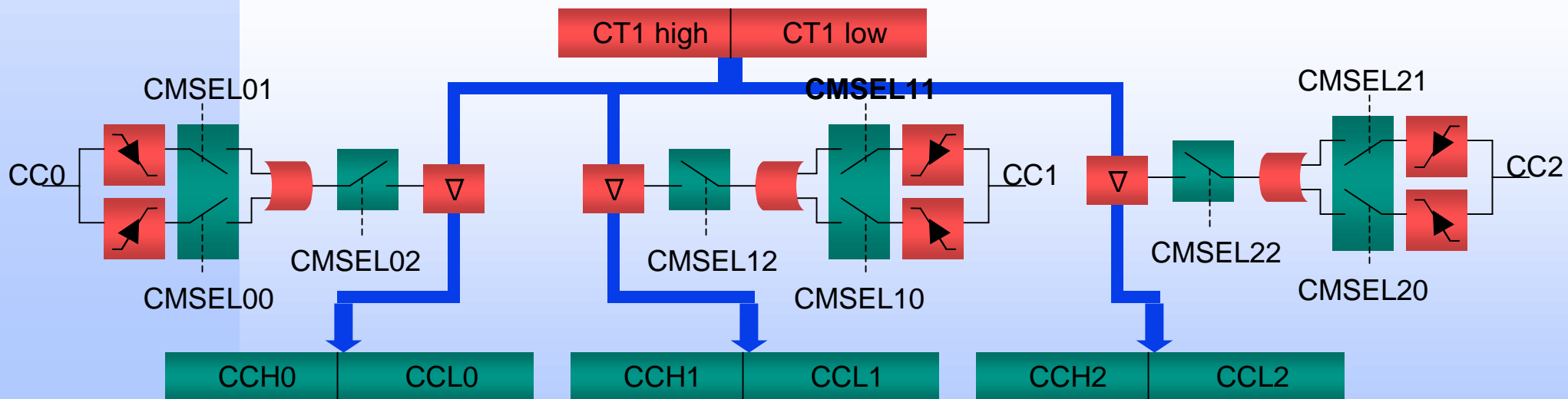
## □ Each Timer can be stopped, started and reset by 2 bits

- CTxR - Compare Timer Run bit
  - When set, the timer runs
  - When cleared the timer stops
- CTxRES - Compare Timer Reset bit
  - When set, the timer is reset to zero
  - When cleared the timer is not reset to zero

# Capture / Compare Unit

## Capture Mode

- ❑ **Compare Timer 1 (16 bit) can perform a 3 channel Capture.**
  - Triggered by external pins (CC0, CC1, and CC2)
    - Falling edge, rising edge, or both
  - CT1 can be operating in Mode 0 or Mode 1 (Edge or Center aligned)
  - The channels can be individually programmed
    - Unused channel(s) can be used for other functions (e.g. Compare functions)



# Capture / Compare Unit

## Compare Modes of CT1

---

### ❑ **CT1 Compare features**

- CT1 can operate in Mode 1 or Mode 2 (center or edge aligned)
- Polarity of compare outputs can be individually programmed

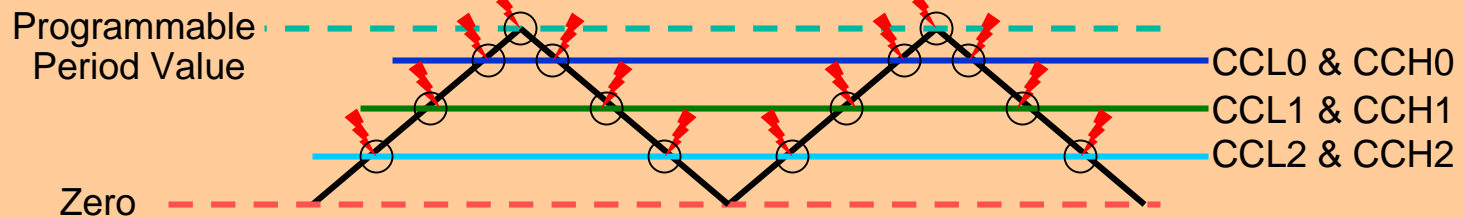
### ❑ **CT1 Compare Modes**

- Mode 0: 3 channel compare with no output
  - Good for periodic interrupts
- Mode 1: 3 channel compare with output signal generation (and interrupts)
  - Good for simple PWM generation
- Mode 2: 6 channel compare
  - 3 independent compare events (like Mode 2) with 3 extra identical or inverted outputs
- Mode 3: 6 channel compare with non-overlapping transitions
  - Same as Mode 3, except extra outputs have non overlapping transitions.
  - Excellent for 3 phase Inverter Control

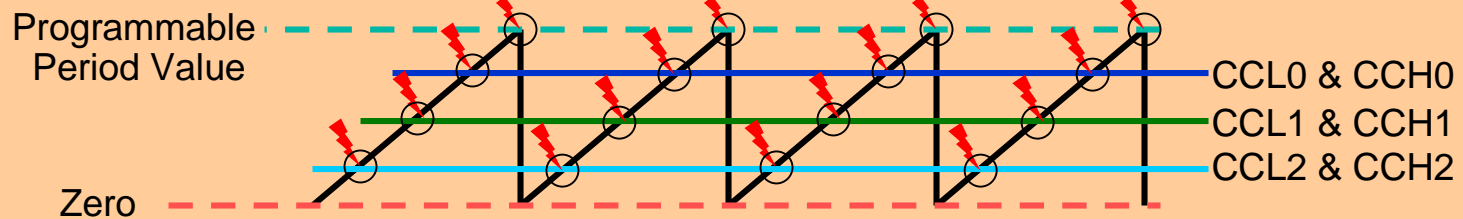
# Capture / Compare Unit

## Compare Modes of CT1 (cont.)

### □ Compare Mode 0



CT1 in Mode 1



CT1 in Mode 0

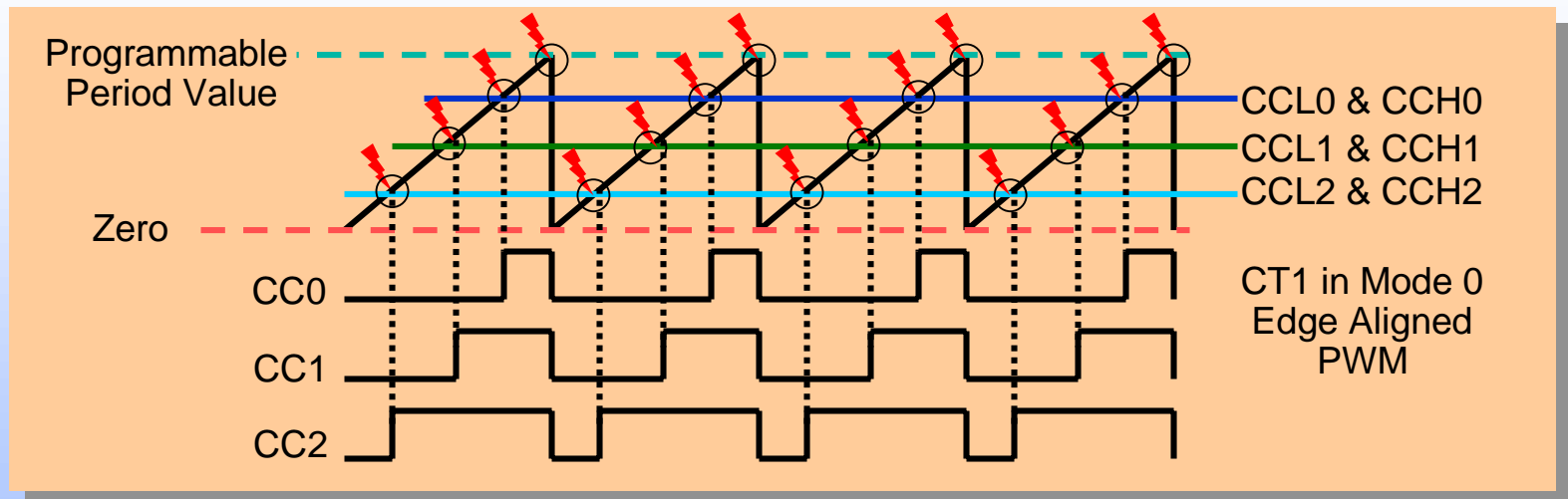
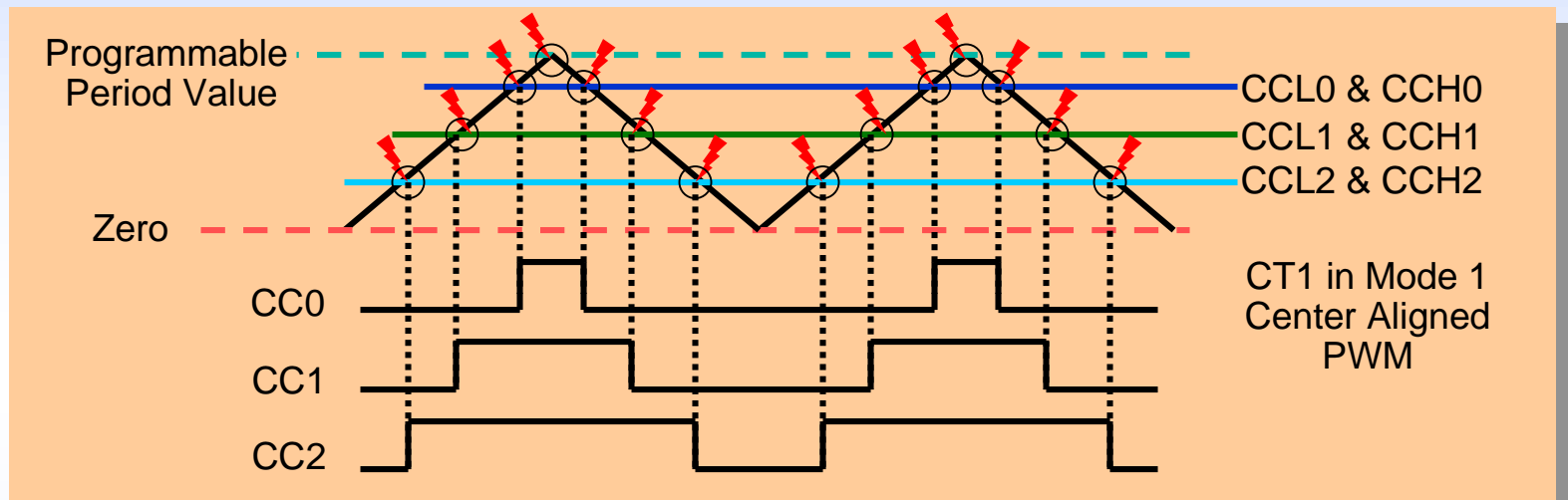
⚡ = Interrupt Generation



# Capture / Compare Unit

## Compare Modes of CT1 (cont.)

### □ Compare Mode 1

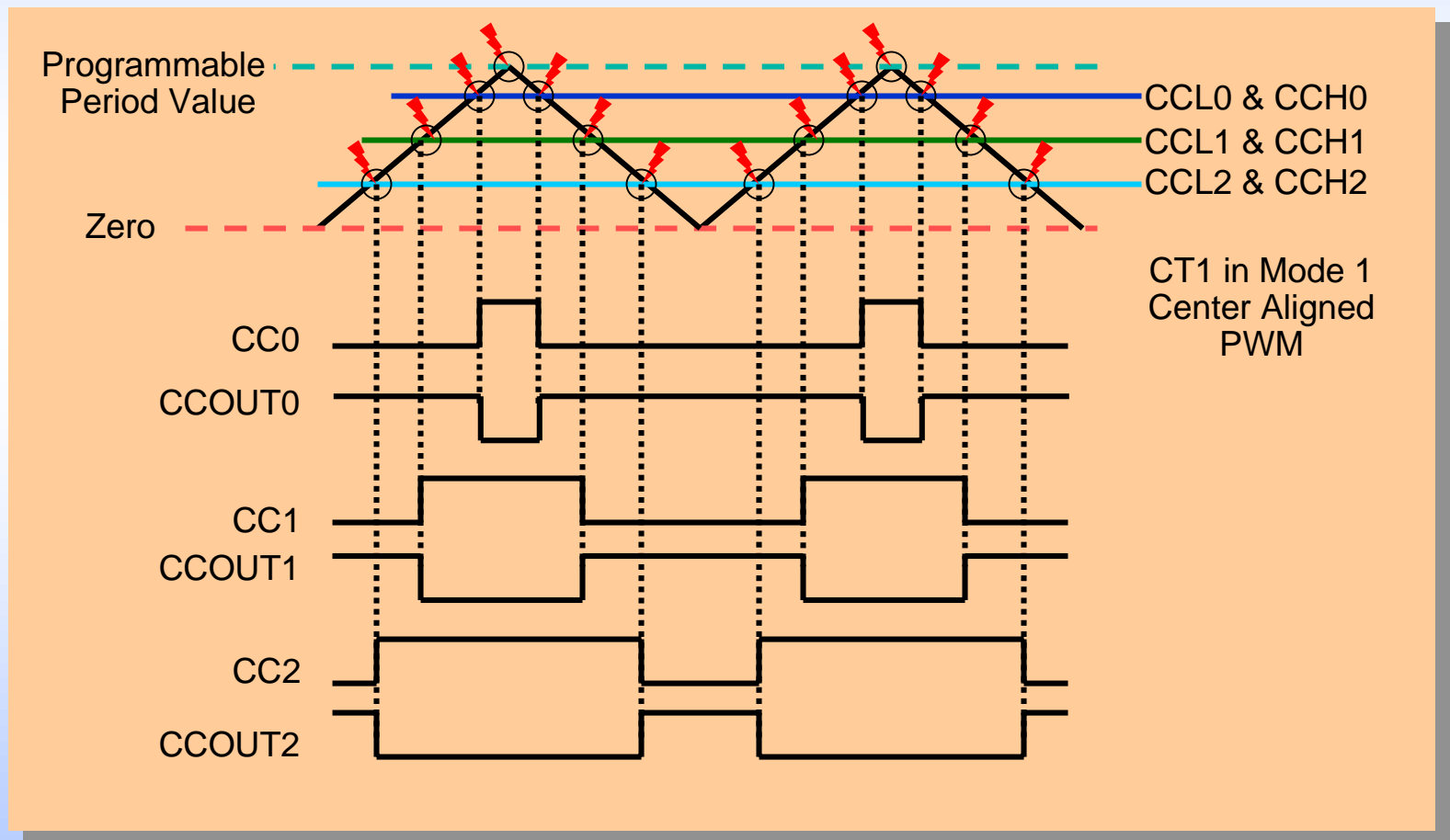


**NOTE: Individual output pins may be inverted.**

# Capture / Compare Unit

## Compare Modes of CT1 (cont.)

### □ Compare Mode 2

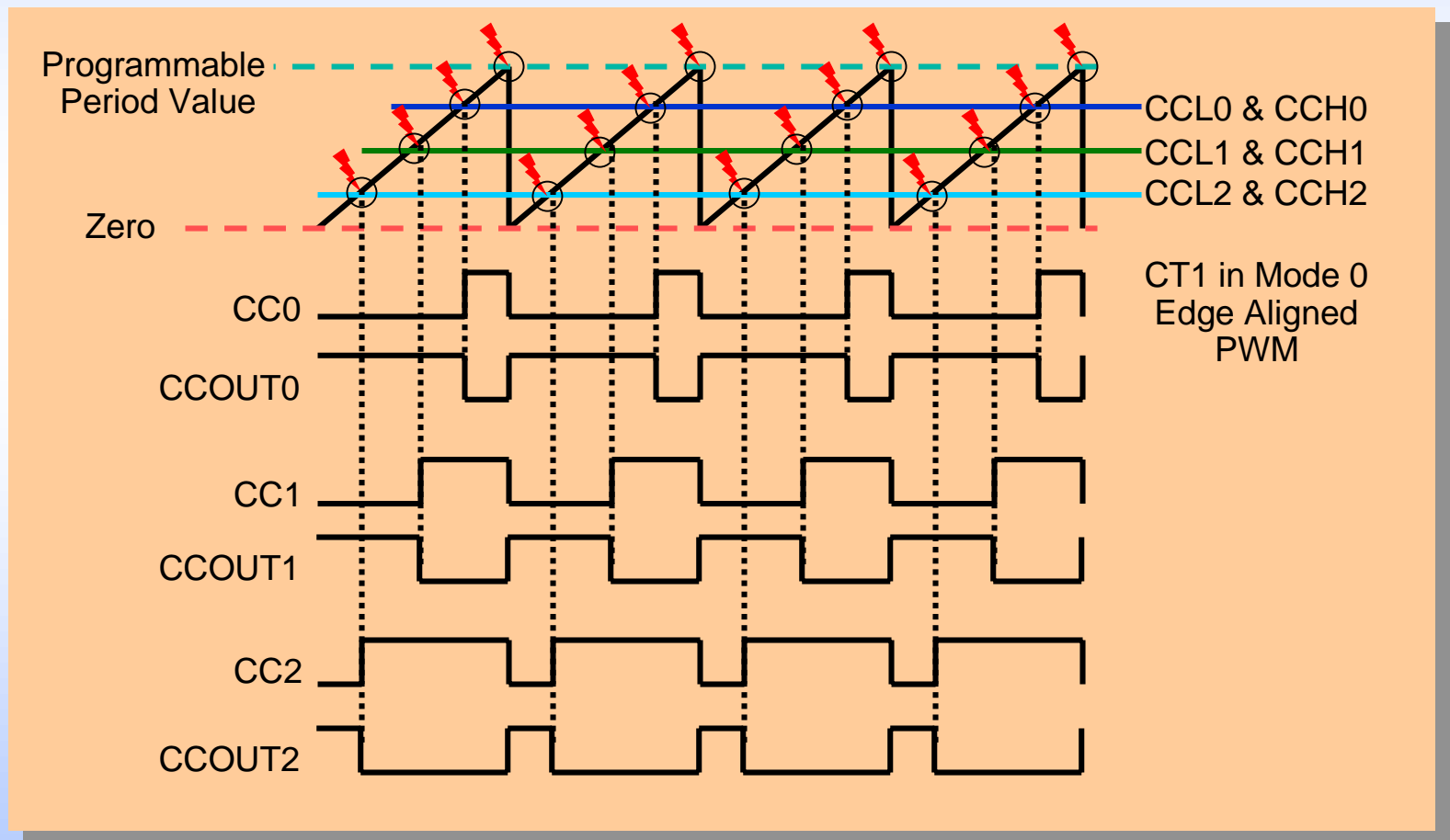


**NOTE: Individual output pins may be inverted.**

# Capture / Compare Unit

## Compare Modes of CT1 (cont.)

### □ Compare Mode 2 (cont.)

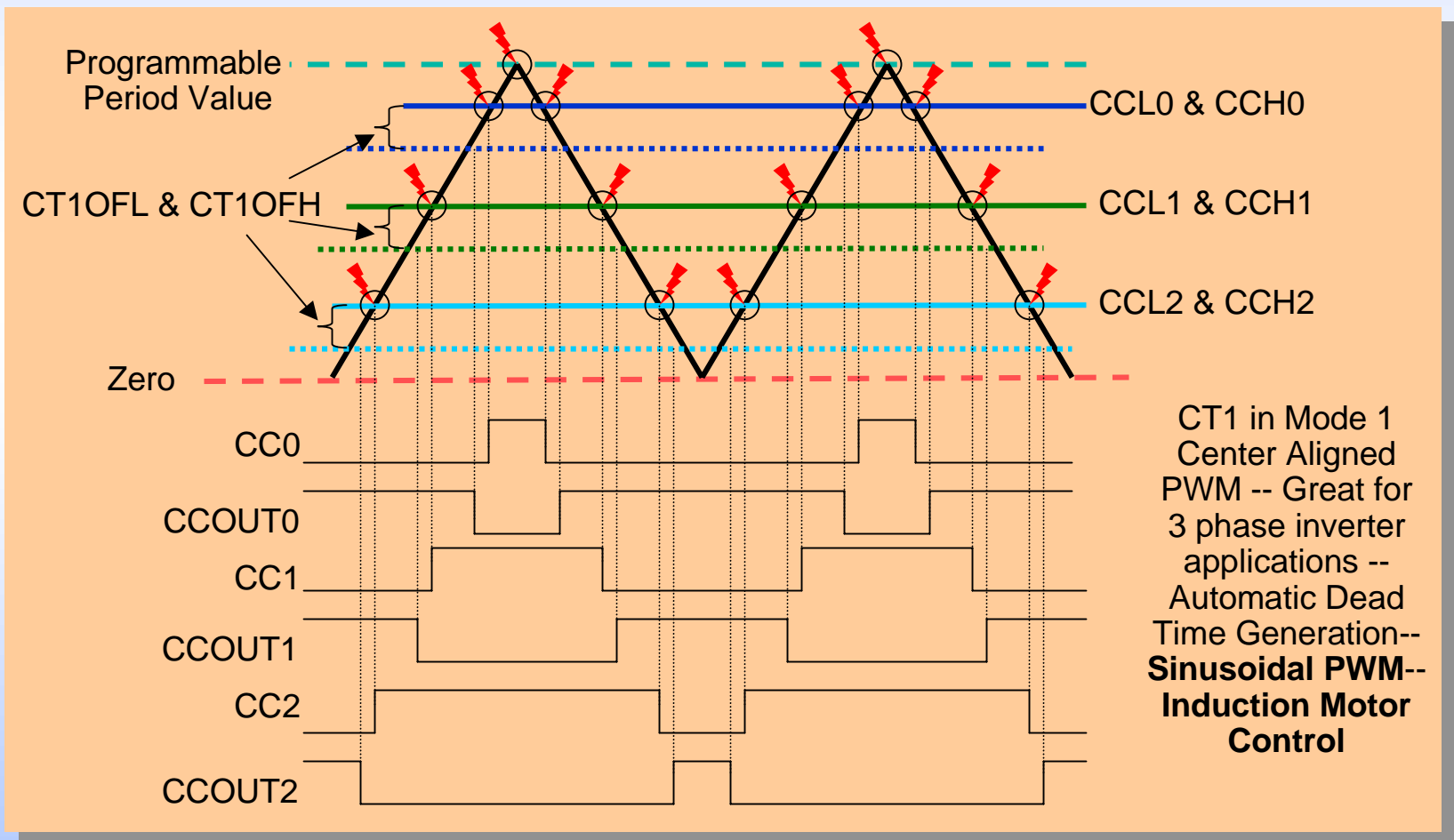


**NOTE: Individual output pins may be inverted.**

# Capture / Compare Unit

## Compare Modes of CT1 (cont.)

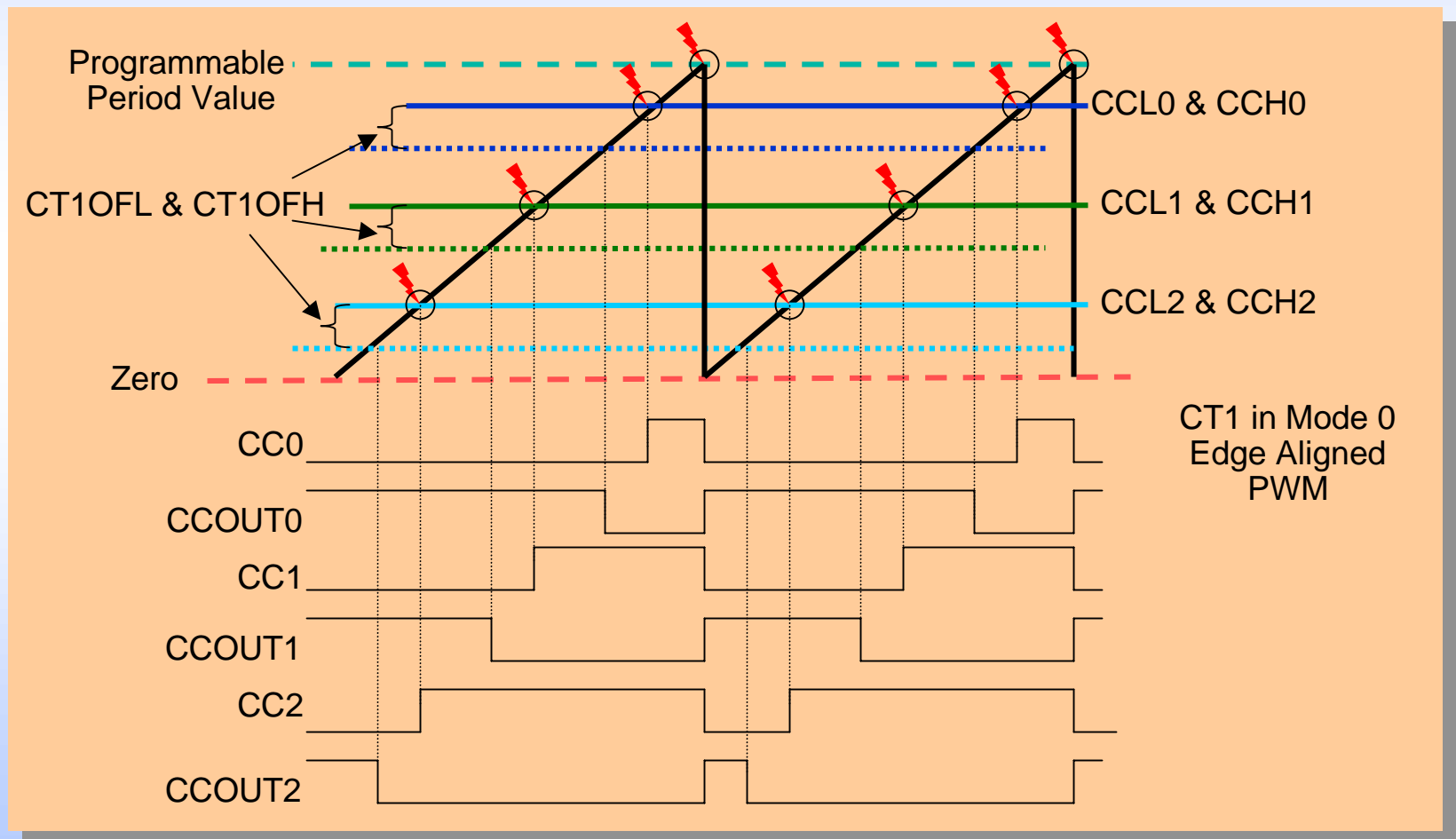
### □ Compare Mode 3 -- non-overlapping transitions



# Capture / Compare Unit

## Compare Modes of CT1 (cont.)

### □ Compare Mode 3 -- non-overlapping transitions (cont.)



**NOTE: Individual output pins may be inverted.**

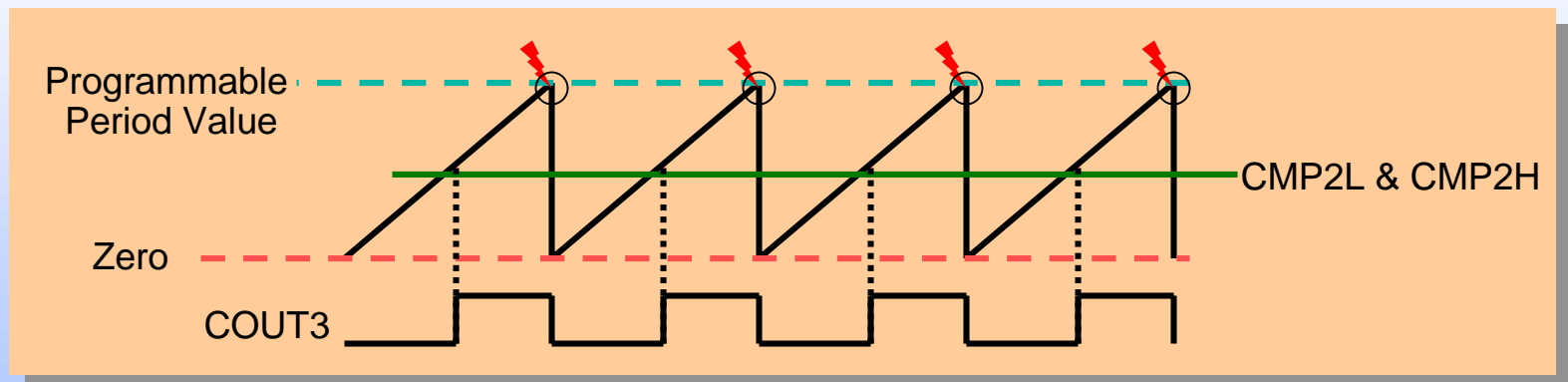
# Capture / Compare Unit

## Compare Timer 2

### □ Compare Timer 2 is a 10-bit timer

- Same operating speed range as CT1
- No Capture Capabilities
- 1 compare channel only, 1 output pin only
- Can operate in Mode 0 (edge aligned) only
- Initial output state (and trap state) is programmable
- Used for Multi-Channel PWM modes and Burst Mode
  - Output can be modulated onto COUTx pins (or CCx and COUTx in Multi-Channel PWM Mode) pins.
  - Good for chopping

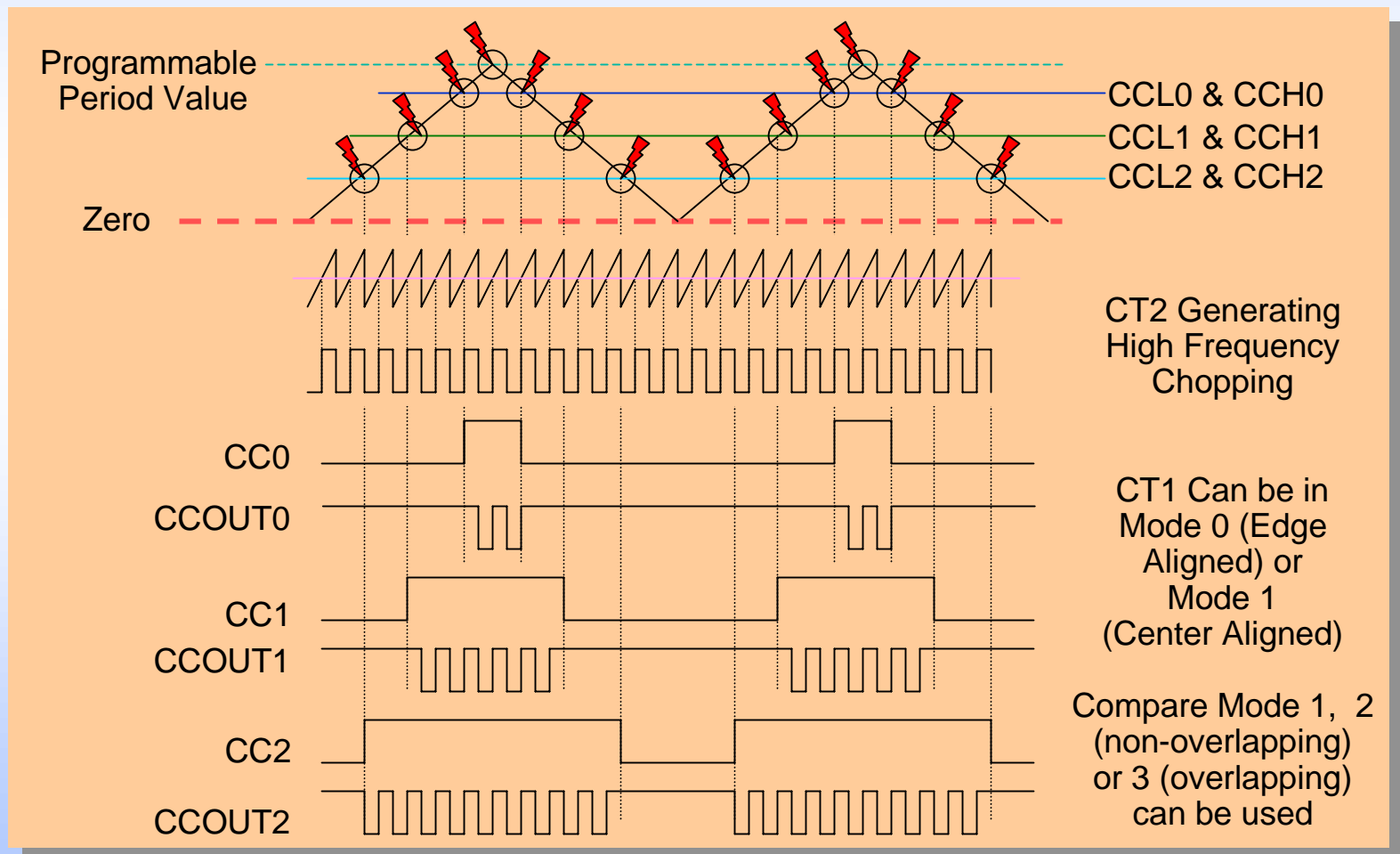
**NOTE: The output pin can be inverted**



# Capture / Compare Unit

## Compare Timer 2 (cont.)

### □ Modulating CT1 outputs with CT2 (Burst Mode)



**NOTE: Individual output pins may be inverted.**

# Capture / Compare Unit

## Motor Control (Multi-Channel PWM) Modes

---

- ❑ **Compare Modes 2 and 3 (previously shown) are great for induction motor control**
- ❑ **Multi-Channel PWM modes are great for synchronous motor control (e.g. Brushless DC Motors)**
  - Block Commutation Mode
    - can control a 3 phase inverter driven motor using position sensors (e.g. Hall Effect sensors)
  - 4-Phase Multi-Channel Mode
    - can drive a unipolar 4 phase motor
  - 5-Phase Multi-Channel Mode
    - can drive a unipolar 5 phase motor
  - 6-Phase Multi-Channel Mode
    - can drive a unipolar 6 phase motor
    - can drive a 3 phase inverter fed motor without position sensors (e.g. 3 phase brushless DC motor using zero-crossing detection for position sensing)
  - In all of these Modes, CT2 can be modulated onto the COUTx outputs. Or CT2 Can be modulated onto the CCx and COUTx outputs.



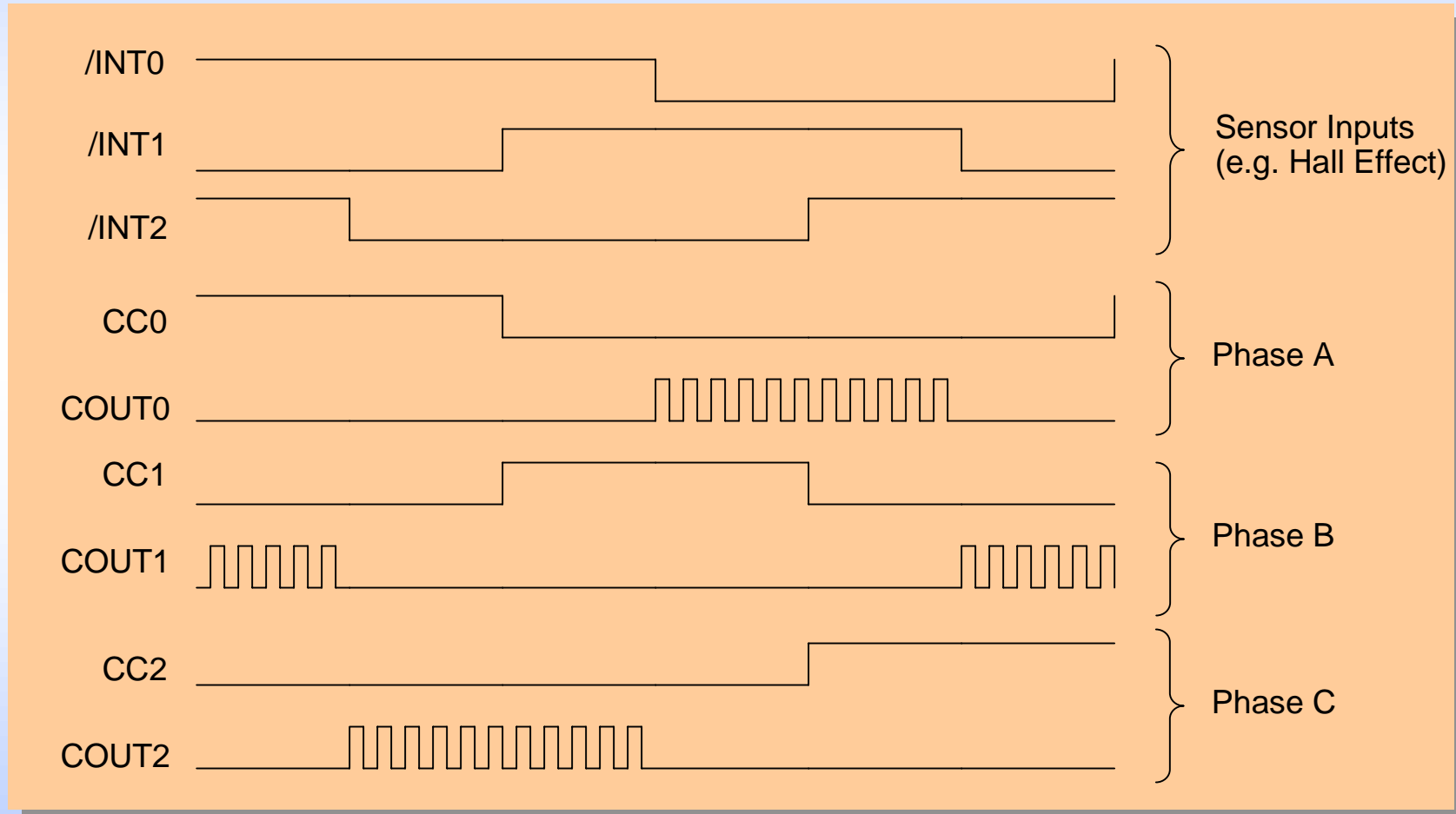
# Capture / Compare Unit Block Commutation Mode

**NOTE: Individual output pins may be inverted.**

**NOTE: Rotation direction may be reversed.**

**NOTE: CT2 can be modulated onto CCx outputs also, or not used at all.**

**NOTE: This mode can be used to control an inverter fed brushless DC motor with minimal CPU load.**



# Capture / Compare Unit

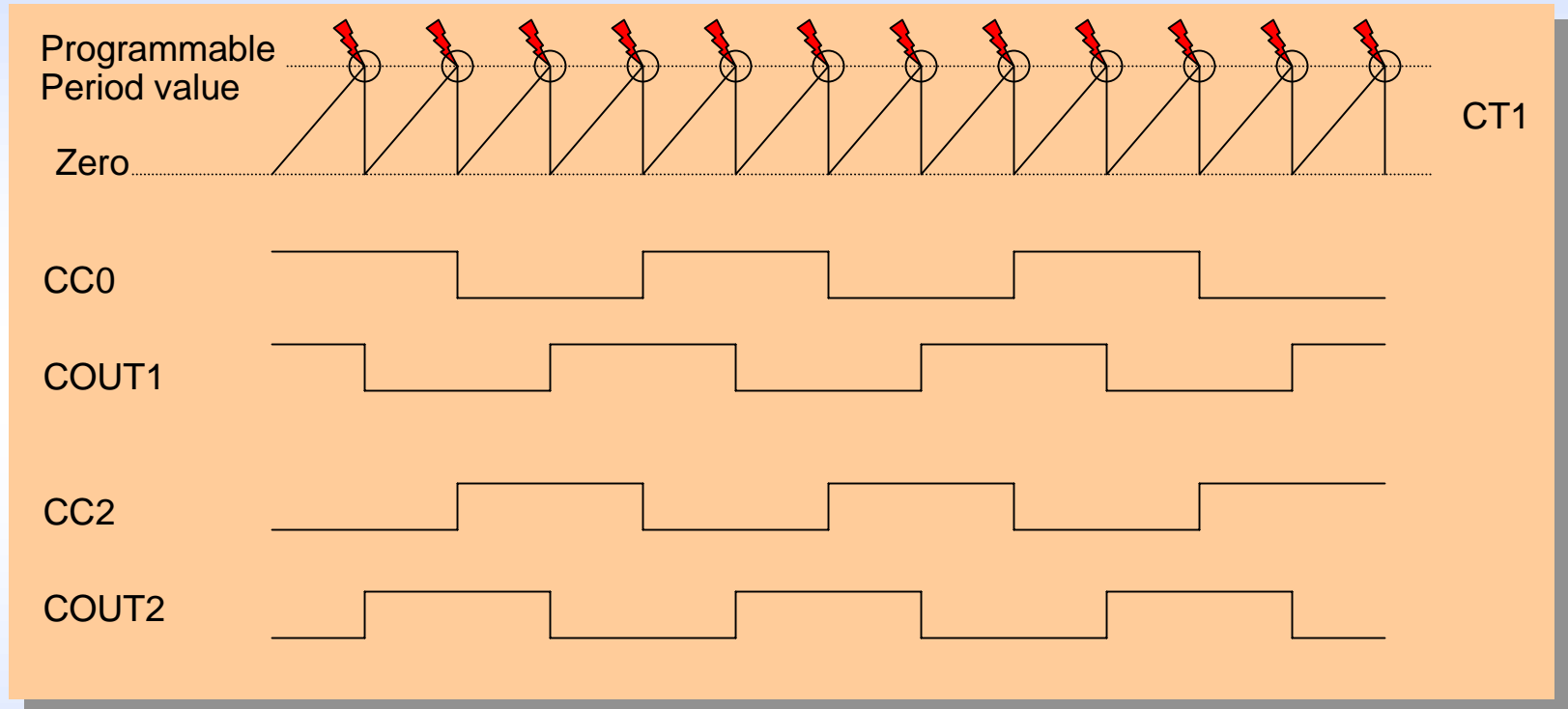
## 4 Phase Multi-Channel PWM Mode

**NOTE: Individual output pins may be inverted.**

**NOTE: Rotation direction may be reversed.**

**NOTE: CT2 can be modulated onto COUTx outputs or CCx outputs and COUTx outputs.**

**NOTE: This mode can be used to control a uni-polar brushless DC motor with minimal CPU load.**



**NOTE: The outputs can be decoupled from CT1. The outputs can switch state in response to software (setting bit NMCS -- Next Multi-Channel State).**

# Capture / Compare Unit

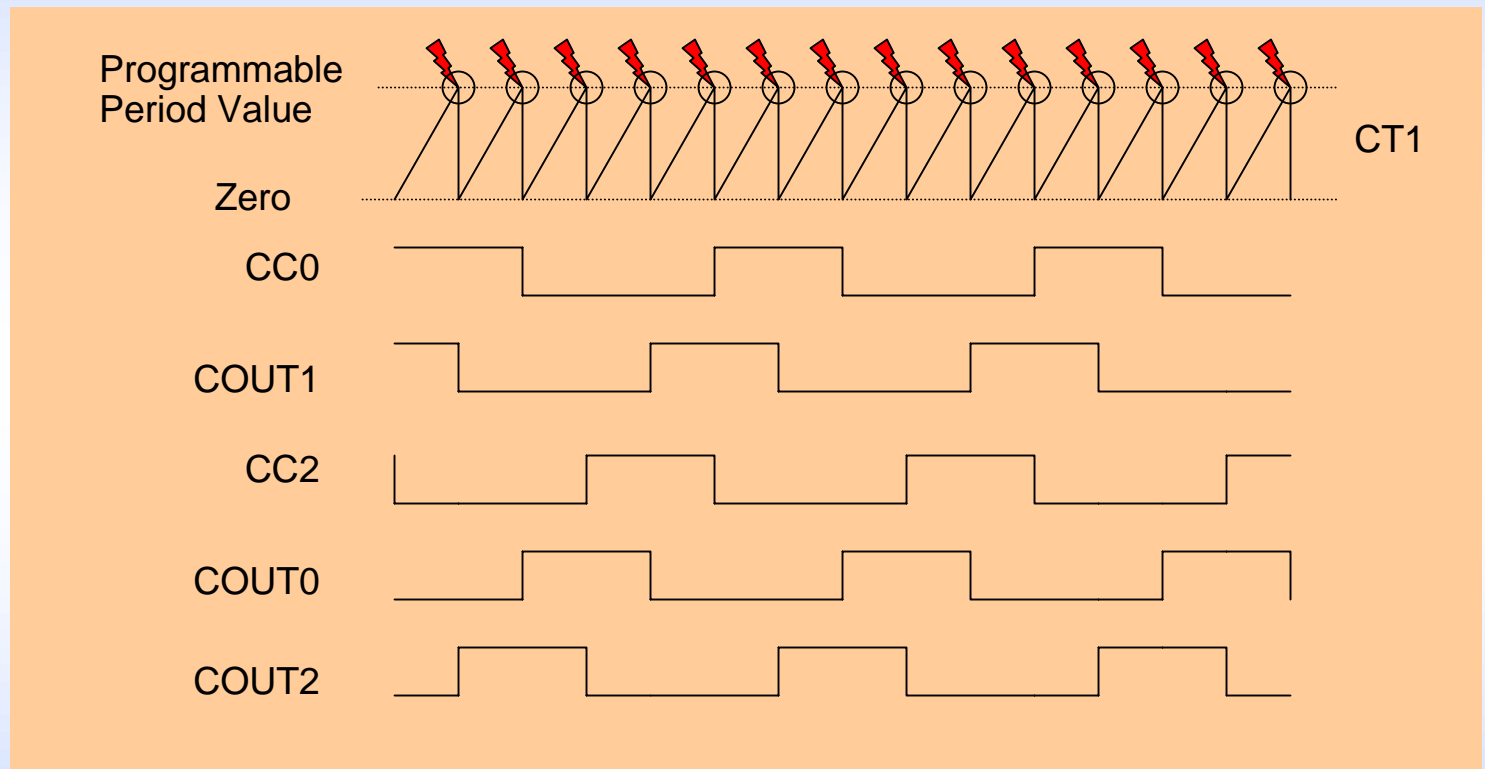
## 5 Phase Multi-Channel PWM Mode

**NOTE:** Individual output pins may be inverted.

**NOTE:** Rotation direction may be reversed.

**NOTE:** CT2 can be modulated onto COUTx outputs or CCx outputs and COUTx outputs.

**NOTE:** This mode can be used to control a uni-polar brushless DC motor with minimal CPU load.



**NOTE:** The outputs can be decoupled from CT1. The outputs can switch state in response to software (setting bit NMCS -- Next Multi-Channel State).

# Capture / Compare Unit

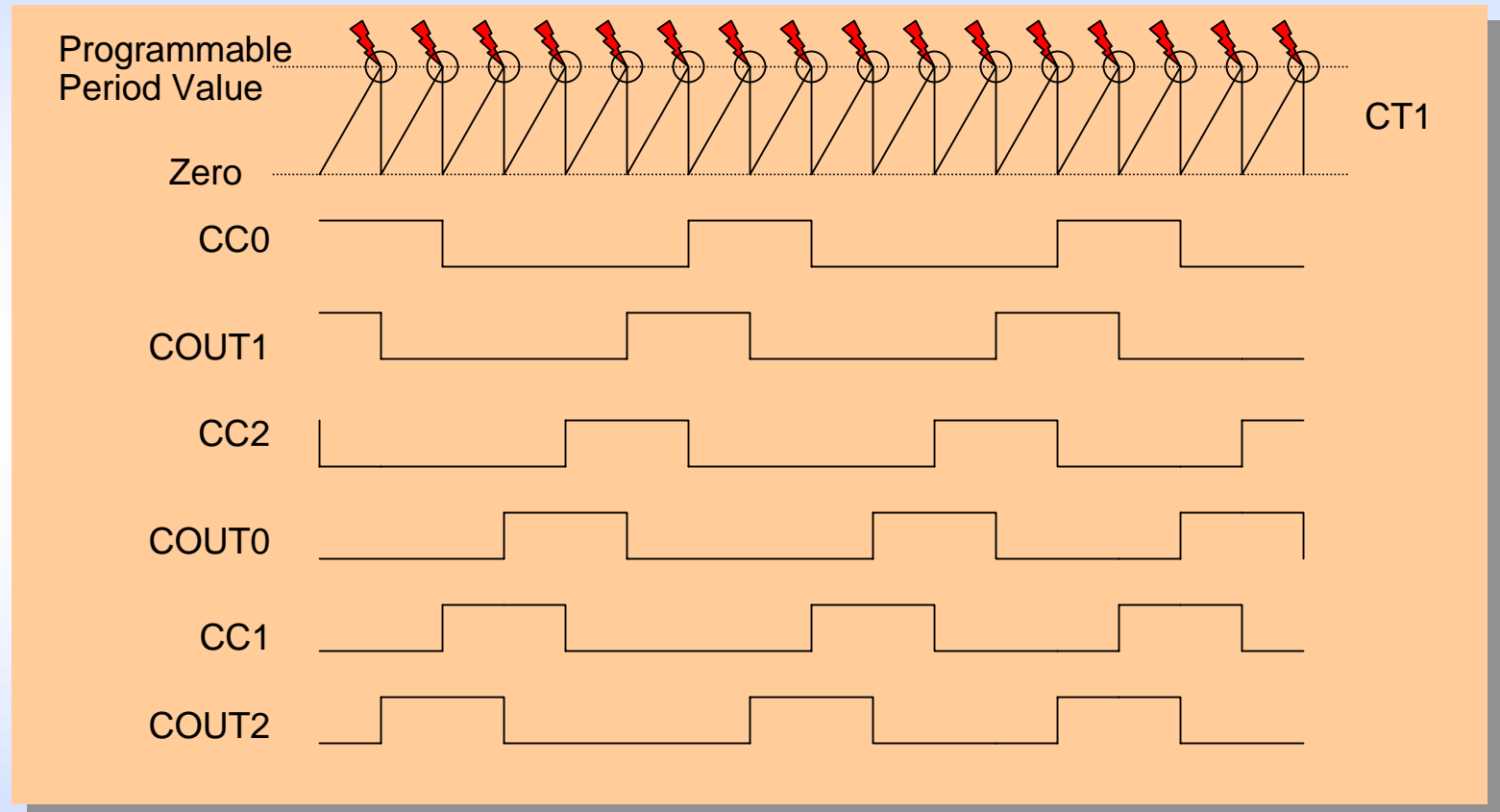
## 6 Phase Multi-Channel PWM Mode

**NOTE:** Individual output pins may be inverted.

**NOTE:** Rotation direction may be reversed.

**NOTE:** CT2 can be modulated onto COUTx outputs or CCx outputs and COUTx outputs.

**NOTE:** This mode can be used to control a uni-polar, 6 phase, brushless DC motor with minimal CPU load. Or a 3 phase bi-polar (inverter fed) brushless DC motor without Hall Effect sensors can be controlled.



**NOTE:** The outputs can be decoupled from CT1. The outputs can switch state in response to software (setting bit NMCS -- Next Multi-Channel State).

# Exercise 4CCU\_1 - Symmetrical PWM Generation - Description

---

## ❑ Objective:



- Generate a 25% duty cycle signal on CC0 and a 75% duty cycle signal on COUT0 (with non-overlapping transitions).
- Generate a 50% duty cycle signal on CC1 and COUT1 (with non-overlapping transitions).
- Generate a 75% duty cycle signal on CC2 and a 25% duty cycle signal on COUT2 (with non-overlapping transitions).
- Use CT1 in Mode 1 (center aligned)

**NOTE:** This example can be used as the starting point for generating 3 phase sinusoidal PWM for inverter applications with automatic dead-time generation. Simply adjust the duty cycles to create any desired sinusoidal output. A lookup table can be used to hold the duty cycle values. There are ApNotes on Sinusoidal PWM and Space Vector Modulation at: <http://www.infineon.com/microcontrollers>

# Exercise 4CCU\_1 - DAVe Configurations

- ❑ **Start DAVe 2.0**



- ❑ **Select “Create a new project” from the Startup Dialog or click** 
- ❑ **Select the microcontroller C504 and click “Create”**  
(if this microcontroller is not on the list, you need to re-install it from the DAVe 2.0 CD ROM\*)
- ❑ **DAVe will create the project**
- ❑ **Save your project by selecting “File | Save” or press** 
  - Browse to directory “c:\hot504\_1\4CCU\_1\”
  - Enter project name: “4CCU\_1”
  - Click “Save”
- ❑ **You will see the C504 block diagram and the Project Settings Window (configuration see next slide)**
- ❑ **To get back to the Project Settings window in case you close it: Select “File | Project Settings”**

# Exercise 4CCU\_1 - DAvE Configurations

---

## ❑ Project Settings:

- General:
  - Controller Type:
    - C504-L40
- System Clock:
  - External Oscillator Frequency: Set to 40 MHz (or the crystal frequency of your KitCON-504 board)
- Close

# Exercise 4CCU\_1 - DAvE Configurations (cont.)

---

## ❑ **Configure the Capture / Compare Unit:**

- **Timer 1:**
  - **Timer 1 Start Control (CT1R):**
    - Start timer 1 after initialization
  - **Timer 1 Operating Mode (CTM):**
    - Center aligned mode: count up/down
  - **Timer 1 Offset**
    - Required edge delay (us): 10
- **Channels:**
  - **Configure Channel 0:**
    - **Capture Compare Mode Selection (CMSELx0,1,2):**
      - Compare output on pin CC0 and COUT0
    - **Output CCx Initial Value (CCxI):**
      - The passive output level on CC0 is low
    - **Output COUTx Initial Value (COUTxI)**
      - The passive output level on COUT0 is high
    - **Duty Cycle**
      - Required duty cycle for channel 0 (%): 25
  - **Close**



# Exercise 4CCU\_1 - DAvE Configurations (cont.)

---

- Configure Channel 1:
  - Capture Compare Mode Selection (CMSELx0,1,2):
    - Compare output on pin CC1 and COUT1
  - Output CCx Initial Value (CCxI):
    - The passive output level on CC1 is low
  - Output COUTx Initial Value (COUTxI)
    - The passive output level on COUT1 is high
  - Duty Cycle
    - Required duty cycle for channel 1 (%): 50
- Close
  
- Configure Channel 2:
  - Capture Compare Mode Selection (CMSELx0,1,2):
    - Compare output on pin CC2 and COUT2
  - Output CCx Initial Value (CCxI):
    - The passive output level on CC2 is low
  - Output COUTx Initial Value (COUTxI)
    - The passive output level on COUT2 is high
  - Duty Cycle
    - Required duty cycle for channel 2 (%): 25
- Close

# Exercise 4CCU\_1 - DAvE Configurations (cont.)

---

- Functions
  - CCU\_vInit

- Close

**Generate Code (  )**

**DAvE will show you all the files that he has generated  
(File Viewer is opened automatically)**

# Exercise 4CCU\_1 - uVision Configurations

---

- Start Keil  $\mu$ Vision**
- Create new Project: c:\hot504\_1\4CCU\_1\4CCU\_1.prj**
- Edit Project:**
  - Add MAIN.C (from c:\hot504\_1\4CCU\_1)
  - Add CCU.C (from c:\hot504\_1\4CCU\_1)
  - Open All
  - Save
- Options/Environment Pathspecs...**
  - In most cases the “Automatically determine path specifications” will work, however if you have problems building the project you may have to enter the paths in manually.

# Exercise 4CCU\_1 - uVision Configurations (cont.)

---

## ❑ Edit MAIN.C:

- include endless loop in main():

```
// USER CODE BEGIN (Main,3)
while(1) {};
// USER CODE END
```

## ❑ Build All

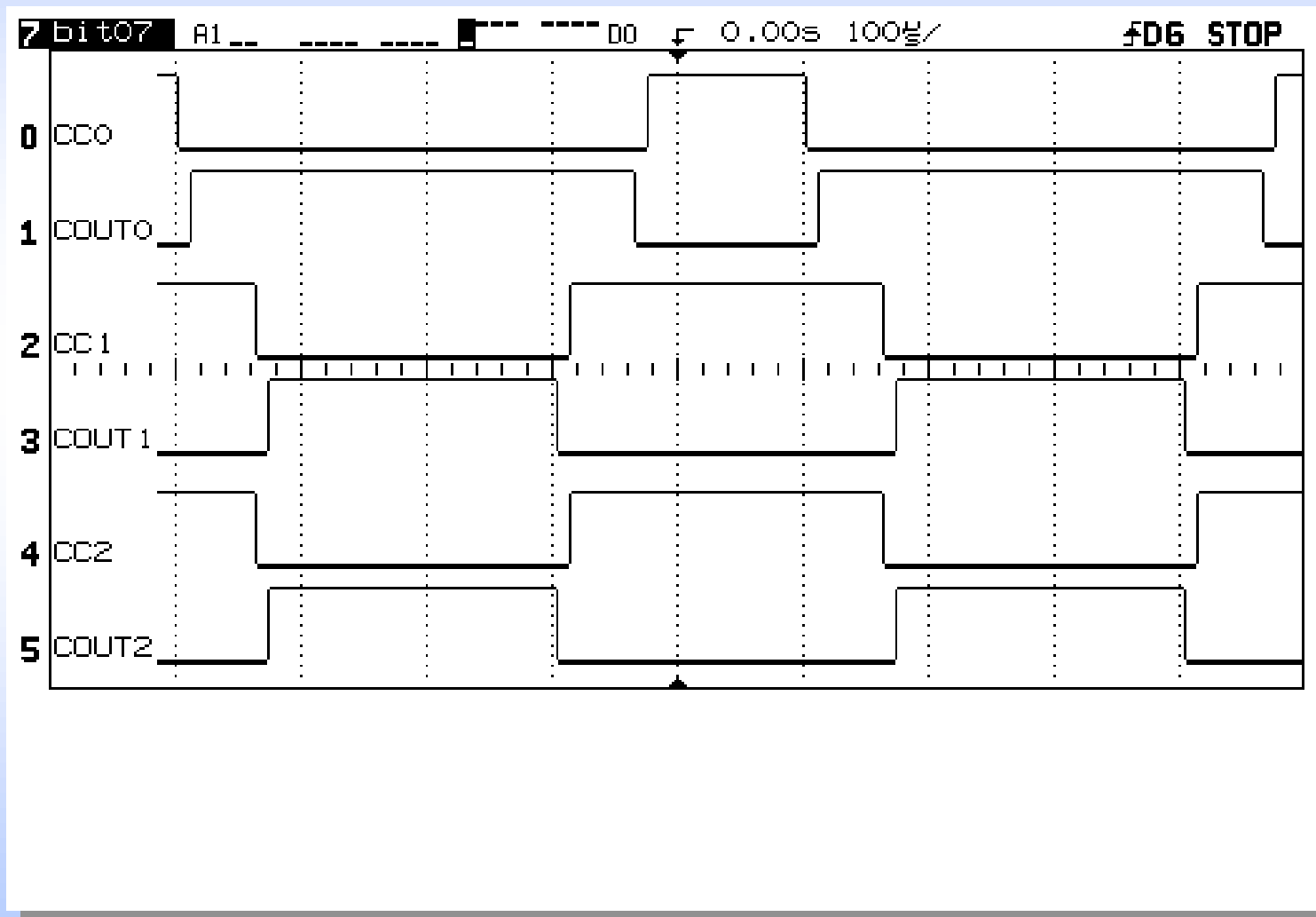
- This will compile and link the project and create an object file and hex file.

# Exercise 4CCU\_1 - Running the Program

---

- **Run dScope Debugger**
- **File | Load CPU Driver: mon51.dll**  
this will establish a connection between the Keil Monitor (located in the flash) and dScope (once this happens dScope becomes tScope - the target debugger)
- **File | Load Object file: Load c:\hot504\_1\4CCU\_1\4CCU\_1**
- **Connect Oscilloscope to P1.2 / CC0, P1.3 / COUT0, P1.4 / CC1, P1.5 / CCOUT1, P1.6 / CC2, P1.7/ CCOUT2 (connector X3 pins 86, 90, 87, 91, 88, 92 respectively)**
- **Hit Go!**
- **When you are finished viewing the output, push the RESET button on the KitCON-504**

# Exercise 4CCU\_1 - Screenshot



# Exercise 4CCU\_2 - Synchronous Motor PWM Generation - Description

---

- ❑ **Generate PWM for a 6 phase unipolar (or 3 phase bi-polar) synchronous motor (e.g. Brushless DC motor).**
  - Use 6 phase Multi-channel PWM mode
  - Use CT1 to control pulse width (active phase)
  - Use CT2 to control chopping (duty cycle during active phase)

**Note:** This example applies the chopping waveform generated by CT2 to all 6 compare outputs. There is also the option of applying the chopping waveform to only the COUTx outputs. Chopping only COUTx outputs would reduce switching losses when controlling a 3 phase inverter fed brushless DC motor.



**Note:** Block commutation mode can be used to control 3 phase brushless DC motors using rotor positions sensors (e.g. Hall-Effect sensors). 6 phase multi-channel PWM mode can be used to control 6 phase unipolar motors, or 3 phase bi-polar (inverter fed) motors without rotor position sensors, since CT1 can be used to implement a phase delay (e.g. 30 electrical degree delay needed for zero crossing detection).

# Exercise 4CCU\_2 - DAVe Configurations

---

- ❑ **Start DAVe 2.0**



- ❑ **Select “Create a new project” from the Startup Dialog or click** 
- ❑ **Select the microcontroller C504 and click “Create”**  
(if this microcontroller is not on the list, you need to re-install it from the DAVe 2.0 CD ROM\*)
- ❑ **DAVe will create the project**
- ❑ **Save your project by selecting “File | Save” or press** 
  - Browse to directory “c:\hot504\_1\4CCU\_2\”
  - Enter project name: “4CCU\_2”
  - Click “Save”
- ❑ **You will see the C504 block diagram and the Project Settings Window (configuration see next slide)**
- ❑ **To get back to the Project Settings window in case you close it: Select “File | Project Settings”**



# Exercise 4CCU\_2 - DAvE Configurations

---

## ❑ Project Settings:

- General:
  - Controller Type:
    - C504-L40
- System Clock:
  - External Oscillator Frequency: Set to 40 MHz (or the crystal frequency of your KitCON-504 board)
- Close

# Exercise 4CCU\_2 - DAvE Configurations (cont.)

---

## ❑ **Configure the Capture / Compare Unit:**

- **Multi-Channel**
  - Multi-Channel Enable Bit
    - Enable multi-channel PWM modes (BCEN)
  - Multi-Channel PWM Mode (PWM0, PWM1)
    - 6-phase multi-channel PWM mode
  - Multi-Channel PWM Mode Output Pattern (BCM0 / 1)
    - Rotate right mode
  - Machine Polarity (BCMP)
    - All enabled compare outputs COUTn and CCn are switched to the T2 output signal during their active phase
  
- **Timer 1:**
  - Timer 1 Start Control (CT1R):
    - Start timer 1 after initialization

# Exercise 4CCU\_2 - DAvE Configurations (cont.)

---

- Timer 2:
  - Timer 2 Start Control (CT2R)
    - Start timer 2 after initialization
  - COUT3 Inversion Control (COUTXI)
    - T2's output signal is used to modulate the compare outputs COUTx in burst or multi-channel mode (x=0..2)
- Channels:
  - Configure Channel 0:
    - Output CCx Initial Value (CCxI)
      - The passive output level on CC0 is low
    - Output COUTx Initial Value (COUTxI)
      - The passive output level on COUT0 is low
    - Duty Cycle:
      - Required duty cycle for channel 0 (%): 100
    - Close


# Exercise 4CCU\_2 - DAvE Configurations (cont.)

---

- Channels:
  - Configure Channel 1:
    - Output CCx Initial Value (CCxI)
      - The passive output level on CC1 is low
    - Output COUTx Initial Value (COUTxI)
      - The passive output level on COUT1 is low
    - Duty Cycle:
      - Required duty cycle for channel 1 (%): 100
    - Close
- Channels:
  - Configure Channel 2:
    - Output CCx Initial Value (CCxI)
      - The passive output level on CC2 is low
    - Output COUTx Initial Value (COUTxI)
      - The passive output level on COUT2 is low
    - Duty Cycle:
      - Required duty cycle for channel 2 (%): 100
    - Close

# Exercise 4CCU\_2 - DAVe Configurations(cont.)

---

- Channels:
    - Configure Channel 3:
      - Enable Timer 2 Output (ECT20)
        - When timer 2 is running, timer T2 output COUT3 is enabled and outputs the PWM signal of the 10-bit compare channel
      - Output COUT3 Initial Value (COUT3I)
        - The passive output level on COUT3 is low
      - Duty Cycle:
        - Required duty cycle for channel 3 (%): 50
      - Save & Close
  - Functions
    - CCU\_vInit
  - Save & Close
- Generate Code (  )**
- DAvE will show you all the files that he has generated (File Viewer is opened automatically)**

# Exercise 4CCU\_2 - uVision Configurations

---

- Start Keil  $\mu$ Vision**
  
- Create new Project: c:\hot504\_1\4CCU\_2\4CCU\_2.prj**
  
- Edit Project:**
  - Add MAIN.C (from c:\hot504\_1\4CCU\_2)
  - Add CCU.C (from c:\hot504\_1\4CCU\_2)
  - Open All
  - Save
  
- Options/Environment Pathspecs...**
  - In most cases the “Automatically determine path specifications” will work, however if you have problems building the project you may have to enter the paths in manually.

# Exercise 4CCU\_2 - uVision Configurations (cont.)

---

## ❑ Edit MAIN.C:

- include endless loop in main():

```
// USER CODE BEGIN (Main,3)
while(1) {};
// USER CODE END
```

## ❑ Build All

- This will compile and link the project and create an object file and hex file.

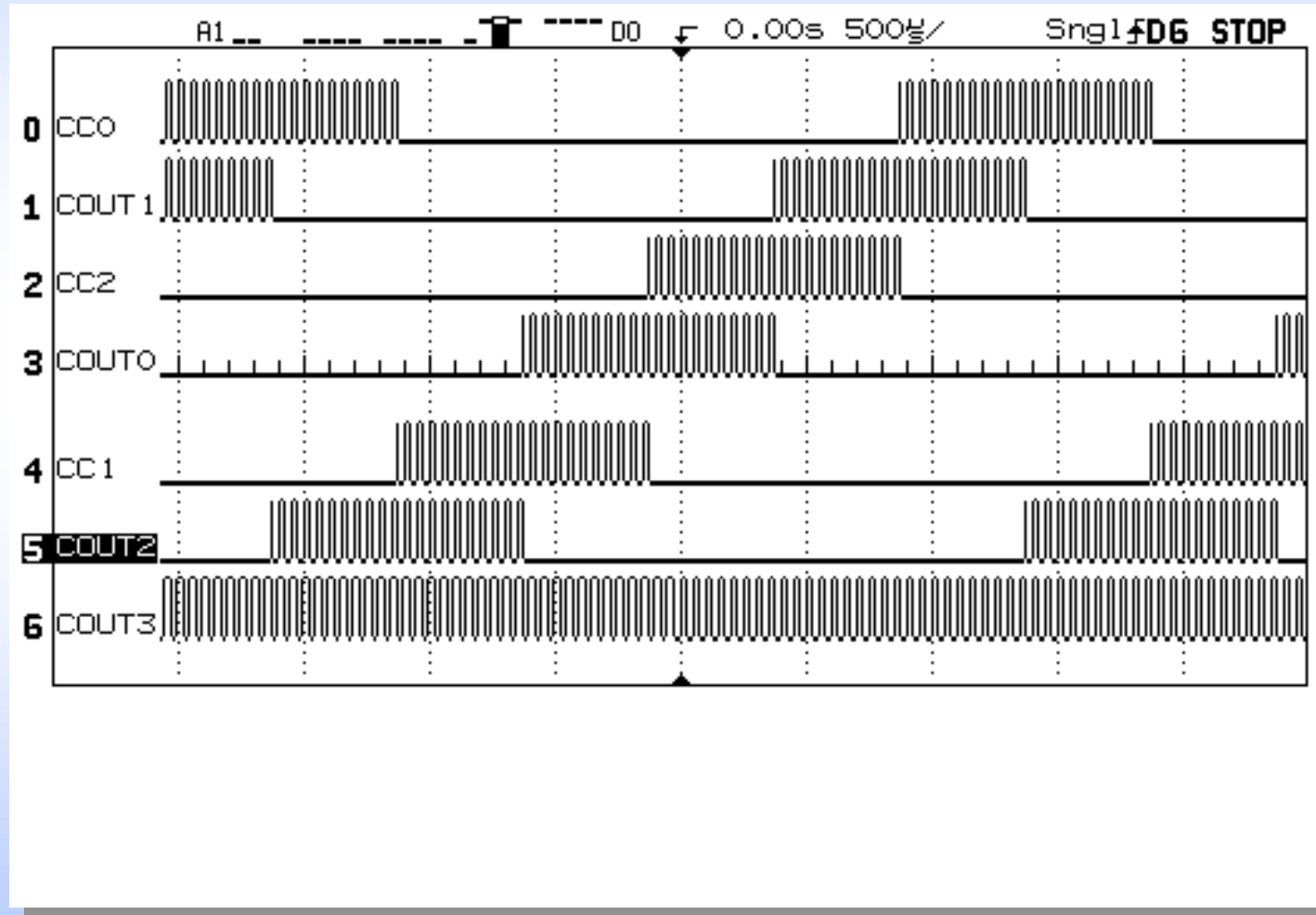
# Exercise 4CCU\_2 - Running the Program

---

- **Run dScope Debugger**
  
- **File | Load CPU Driver: mon51.dll**  
this will establish a connection between the Keil Monitor (located in the flash) and dScope (once this happens dScope becomes tScope - the target debugger)
  
- **File | Load Object file: Load c:\hot504\_1\4CCU\_2\4CCU\_2**
  
- **Connect Oscilloscope to P1.2 / CC0, P1.3 / COUT0, P1.4 / CC1, P1.5 / CCOUT1, P1.6 / CC2, P1.7/ CCOUT2, COUT3 (connector X3 pins 86, 90, 87, 91, 88, 92, 60 respectively)**
  
- **Hit Go!**
  
- **When you are finished viewing the output, push the RESET button on the KitCON-504**



# Exercise 4CCU\_2 - Screenshot



# Watchdog Timer

---

- ❑ **15-Bit timer overflow results in:**
  - Software reset
  - Sets WDTS flag
  
- ❑ **Programmable input clock**
  
- ❑ **High 7-bits reload register**
  
- ❑ **Timer period from 256  $\mu$ s to 0.55 s @24 MHz**
  
- ❑ **Can be reloaded with a special instruction sequence**
  - Set Bit WDT
  - Set Bit SWDT

# Watchdog Timer Block Diagram

