EXAMPLES

Examples are mainly intended for the students to get started quickly with the basic concepts, programming language, and hex/binary conventions. The following examples are available:

Word file	ASM file(s)			
Example_1	Ex1.asm			
Example_2	Ex2.asm			
Example_3	Ex3.asm			
Example_4	Ex4.asm			
Example_5	Ex5.asm			
Example_6	Ex6.asm			
Example_7	Ex7.asm			
Example_8	Ex8.asm			
Example_9	Ex9.asm			
Example_10	Ex10.asm			
Example_delay	Ex_Delay.asm			
Example_sq_wave	Sq_wav1.asm	Sq_wav1.asm		
Example_buttons_bx	Buttons_bx.asm			
Example_SCI_transm	SCI_transm.asm			
Example_SCI_recept	SCI_recept.asm			
Example_IC	Ex_IC.asm			
Example_OC	Ex_OC.asm			
Example_RPM_1	Ex_RPM_1.asm			
Example_RPM_2	Ex_RPM_2.asm	HEX_BCD.asm		
Example_Step	Ex_Step.asm			
Example_Long_Delay	Ex_Long_Delay.asm			
Example_Sort	Ex_Sort.asm			
Example_AD_convert	Ex_AD_1.asm	Ex_AD_2.asm	Ex_AD_3.asm	Ex_AD_4.asm

An overview of the examples is presented next.

EXAMPLES OVERVIEW

#	Concepts
Example_1	• Familiarize the user with the THRSim11 simulator environment • Introduce the user to the syntax and concepts of Assembly (.asm) language • Familiarize the user the way arithmetic operations are handled by the microcontroller • Instruct the user to interpret the List (.LST) file. • Teach the user to perform the simulation and follow the step-by-step results. • Introduce the LDAA, LDAB and ABA operations • Introduce *inherent* mode • Introduce *immediate mode*
Example_2	• Familiarize the user with the THRSim11 simulator environment • Introduce the user to the syntax and concepts of Assembly (.asm) language • Familiarize the user the way arithmetic operations are handled by the microcontroller • Instruct the user to interpret the List (.LST) file. • Teach the user to perform the simulation and follow the step-by-step results. • Introduce ADDA operation
Example_3	• Review the principles of the decimal numbers, i.e., in the range 0 – 9 • Review the decimal carry concept, i.e., when a result is outside the 0 – 9 range. • Prepare the ground for hex arithmetic
Example_4	• Introduce *hex numbers* and *hex symbols*, a, b, c, d, e, f. • Introduce *single precision* and *double precision* registers • Introduce *hex arithmetic*.
Example_5	• Review 2-digit decimal arithmetic with carry to prepare the ground for 2-digit hex arithmetic with carry.
Example_6	• Introduce *hex arithmetic with carry* • Facilitate comparison with the decimal case by using the same numbers as in Example 5

Example_7	• Introduce 2-digit *hex arithmetic with carry* • Introduce the 'lost carry' concept • Introduce unsigned hex overflow • Introduce incrementation (INCA)
Example_8	• Introduce decrementation (DECA) • Review decimal borrow concepts using decrementation • Introduce hex borrow using decrementation • Introduce 2's complement negative hex numbers as decrementation of zero with free borrow • Show the decimal equivalent of the 2-digit hex decrementation of zero
Example_9	• Introduce hex subtraction • Introduce hex subtraction with negative results using the free borrow concept. • Show the decimal equivalent of hex subtraction with negative results using free borrow.
Example_10	• Introduce 8-bit binary numbers • Introduce connection between 8-bit binary numbers and 2-digit hex numbers. • Introduce 8-bit binary arithmetic • Introduce binary 8-bit logic • Introduce masking
Example_ delay	• Review the use of branching • Illustrate a delay (wait) program • Demonstrate a flowchart and its relation with the program tasks list and program code • Demonstrate the calculation of clock cycles from the clock cycles of each operation • Demonstrate the calculation of MCU time from the clock cycles
Example_ sq_wave	• Review the use parallel ports • Illustrate the generation of a square wave using Port B output and a waiting sequence • Demonstrate the use of a subroutine to shorten the length of a program • Examine the relation between displayed simulation time and clock cycles in THRSim11 simulator

Example_ buttons_bx	• Review the use parallel ports • Illustrate the use of Port C for input and output • Introduce the concept of shifting register and memory content • Practice the use of the THRSim11 IO box for input and output
Example_ SCI_transm	• Review the use of serial communication interface (SCI) for transmission • Illustrate the sending of a character through SCI
Example_ SCI_recept	• Review the use of serial communication interface (SCI) for reception • Illustrate the receiving of a character through SCI
Example_IC	• Review the use of MCU Timer function as an Input Capture (IC) device • Review the use of the free running clock, TCNT, and it overflow flag, TOF • Review the use of a input capture clock, TIC1, and its event flag, IC1F • Demonstrated how the selection of signal transition to be captured is made (here rising edge, EDG1A) and that the MCU is only sensitive to that particular transition. • Show the calculation of actual time in μs from the timer readings, T1, T0, and overflow count, NOF.
Example_OC	• Review the use of MCU Timer function as an Output Compare (OC) device • Review the setting of OMx and OLx bits to select a desired OC event (in this example, we set OM3=0, OL3=1, to generate a toggle on the OC3 pin) • Review the detection of the TOCx match with TNCT and the corresponding OC action • Present the correlation between delay, DT, with actual frequency of the square wave. • Explore the accuracy with which frequency can be adjusted • Explore the determination of low and high bounds on the frequencies that can be generated with the MCU
Example_ RPM_1	• Review the use of MCU Timer function as an Input Capture (IC) device • Review the use of the free running clock, TCNT, and it overflow flag, TOF • Review the use of a input capture clock, TIC1, and its event flag, IC1F • Demonstrated how the selection of signal transition to be captured is made (here rising edge, EDG1B) and that the MCU is only sensitive to that particular transition. • Introduce the use of soft masks (here, IC1_MSK, TOF_MSK). • Introduce the concept of layered flowcharting of a program: big-picture with generic descriptions; detailed flowcharts for particular sections of the program.

Example_ RPM_2	• Review the used of two time captures to calculate the time duration of an event • Illustrate the use of scaling factors (100s) to deal with numbers that are larger than the microcontroller word length • Illustrate the handling of overflow counts, with special attention to the situation when the two time captures are on one side and another of a timer overflow (time-change line) • Introduce the method of calculating multiplication through repeated additions and decrementation of a counter • Introduce the use of a conversion subroutine to convert from hex to BCD • Illustrate rotation speed calculation from rotation period. • Illustrate the use of scaling factors (100s) to present the result on a display with limited number of digits.
Example_ Step	• Review the use and control of stepper motors. • Discuss the stepper motor energizing patterns, 1- and 2-phase energizing types, full- and half-step motion. • Introduce the concept of creating quasi-continuous motion through a sequence of steps • Introduce the concept of sequential accessing a finite set of stored patterns through index addressing with a continuously updating pointer. • Illustrate a method to 'load' a single-precision (2-hex) variable into a double precision (4-hex) register. • Introduce the concept of automatic incrementation/decrementation of the addressing pointer with a programmable step size • Discuss the reset actions to be taken when the pointer hits the 'roof' or 'floor'.
Example_ Sort	• Review the use of keystroke commands for controlling a process • Introduce the concept of 'multiple sort' and its 'sequential sort' equivalent
Example_ Long_Delay	• Present a subroutine that can implement a long delay between stepper motor steps • Introduce the concept of how to achieve double precision delays (4-hex) using single precision (2-hex) variables
Example_ AD_convert	• Review the use of A/D conversion function of the M68HC11 microcontroller. • Illustrate various modes of performing A/D conversion.