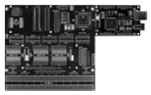


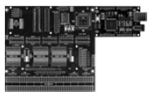
# EEL 3744

## Menu

- Multi-Tasking
  - >Using RTI (a simple Timer System)
    - – Outline the steps in multitasking
    - – "Multitask" with single process
      - ☞Simulate single process
    - – Multitasking example
  - >RTI **and** TOF together for multitasking


See examples on web-site: **`RTI_Step.asm, RTI_1Process.asm, RTI_Mult_Proc.asm, RTI_Mult_Proc2.asm, RTI_Mult_TOF.asm`**

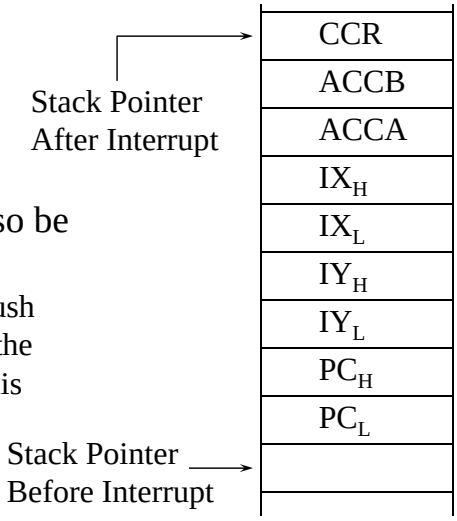


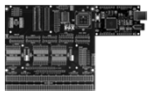


# EEL 3744

## Multi-Tasking using RTI

- Stack Usage by the Interrupt (e.g., RTI)
  - >For the XMEGA, RTC is equivalent, but TC can also be used
    - – Since XMEGA interrupts push nothing other than PC onto the stack, you will need to do this yourself



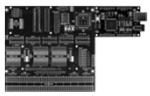


# EEL 3744

## Multi-Tasking using RTI

| P1: Context for P1 includes | P2: Context for P2 includes | ••• | PN: |
|---|---|---|---|
| • Process ID or Name | • Process ID or Name | | • |
| • Starting Address | • Starting Address | | • |
| • Registers A,B,CCR,IX,IY | • Registers A,B,CCR,IX,IY | | • |
| • Regular Stack Pointer (SP) | • Regular Stack Pointer (SP) | | • |
| • Interrupt Stack | • Interrupt Stack | | • |





# EEL 3744

## Multi-Tasking using RTI

| **P1**: Context for **P1** includes |
|---|
| • Process ID or Name |
| • Starting Address |
| • Registers A,B,CCR,IX,IY |
| • Regular Stack Pointer (SP) |
| • Interrupt Stack |

Assume **P1** is running and RTI interrupts

- Inside RTI_ISR the stack contains:

  CCR, B, A, IX, IY & PC **for P1**

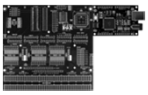
- Let's assume RTI_ISR ***knows*** that PID=1
- Save SP (for P1) into $SP_1$ , i.e., SP→$SP_1$
- Then if RTI_ISR wants to go to P2

  Let SP ← $SP_2$ (SP for P2)

  Change PID to correspond to P2

- RTI_ISR clears RTIF, executes RTI inst.

  This restores the stack of P2



# EEL 3744

## Multi-Tasking using RTI

**P2**: Context for **P2** includes
- Process ID or Name
- Starting Address
- Registers A,B,CCR,IX,IY
- Regular Stack Pointer (SP)
- Interrupt Stack

Now **P2** is running and RTI interrupts

- Inside RTI_ISR the stack contains:

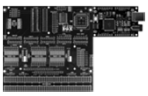
  CCR, B, A, IX, IY & PC **for P2**
- Let's assume RTI_ISR ***knows*** that PID=2
- Save SP (for P2) into $SP_2$ , i.e., SP→$SP_2$
- Then if RTI_ISR wants to go to $P_i$

  Let SP ← $SP_i$ (SP for $P_i$)

  Change PID to correspond to $P_i$
- RTI_ISR clears RTIF, executes RTI inst.

  This restores the stack of Pi





# EEL 3744

## Multi-Tasking using RTI

**PN**: Context for **PN** includes
- Process ID or Name
- Starting Address
- Registers A,B,CCR,IX,IY
- Regular Stack Pointer (SP)
- Interrupt Stack

Assume **PN** is running and RTI interrupts

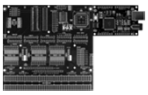
- Inside RTI_ISR the stack contains:

  CCR, B, A, IX, IY & PC **for PN**
- Let's assume RTI_ISR ***knows*** that PID=n
- Save SP (for PN) into $SP_N$ , i.e., SP→$SP_N$
- Then if RTI_ISR wants to go back to P1

  Let SP ← $SP_1$ (SP for P1)

  Change PID to correspond to P1
- RTI_ISR clears RTIF, executes RTI inst.

  This restores the stack of P1



# EEL 3744

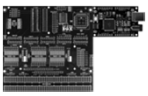
## Multi-Tasking using RTI

**P1**: Context for **P1** includes
- Process ID or Name
- Starting Address
- Registers A,B,CCR,IX,IY
- Regular Stack Pointer (SP)
- Interrupt Stack

**Q**: How do we get things started?

**A**: In the main program:
- Setup RTI interrupt vector ($FFF0) or pseudo vector ($00EE)
- Setup variables & constants
- Create a "dummy" stack for each process:
  CCR, B, A, IX, IY, & PC (entry point for P1)
- Setup RTI system
- Setup any "global" variables
- Enable interrupts
- Jump to the first process you want to run

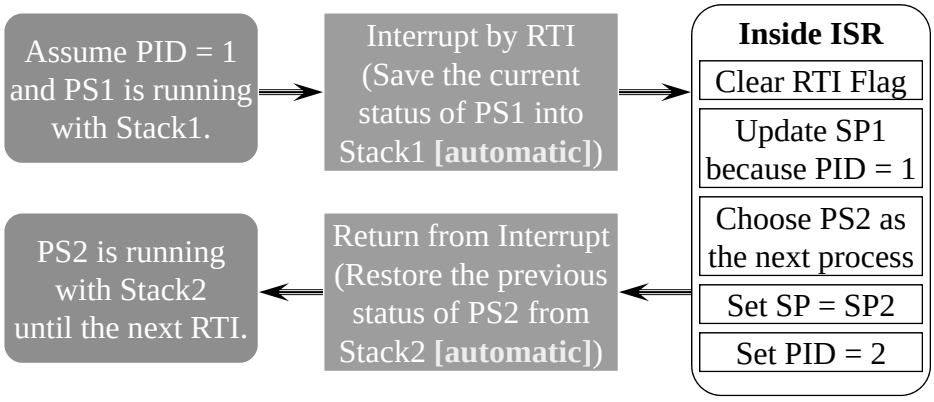


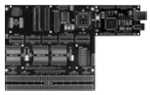


# EEL 3744

## Multi-Tasking using RTI

- Multi-Tasking needs to allocate PID (Process ID), Stack, and Stack Pointer for each Process.





EEL 3744

## Multi-Tasking using RTI

- Multi-Tasking Examples
  - > Outline the steps
  - > Do it with single process
    - > Simulate single process
  - > A multitasking example
  - > Another multitasking example
  - > RTI **with** TOF (i.e., two interrupts working together):

`RTI_Step.asm`

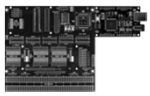
`RTI_1Process.asm`

`RTI_Mult_Proc.asm`

`RTI_Mult_Proc2.asm`

`RTI_Mult_TOF.asm`





EEL 3744

# *The End!*