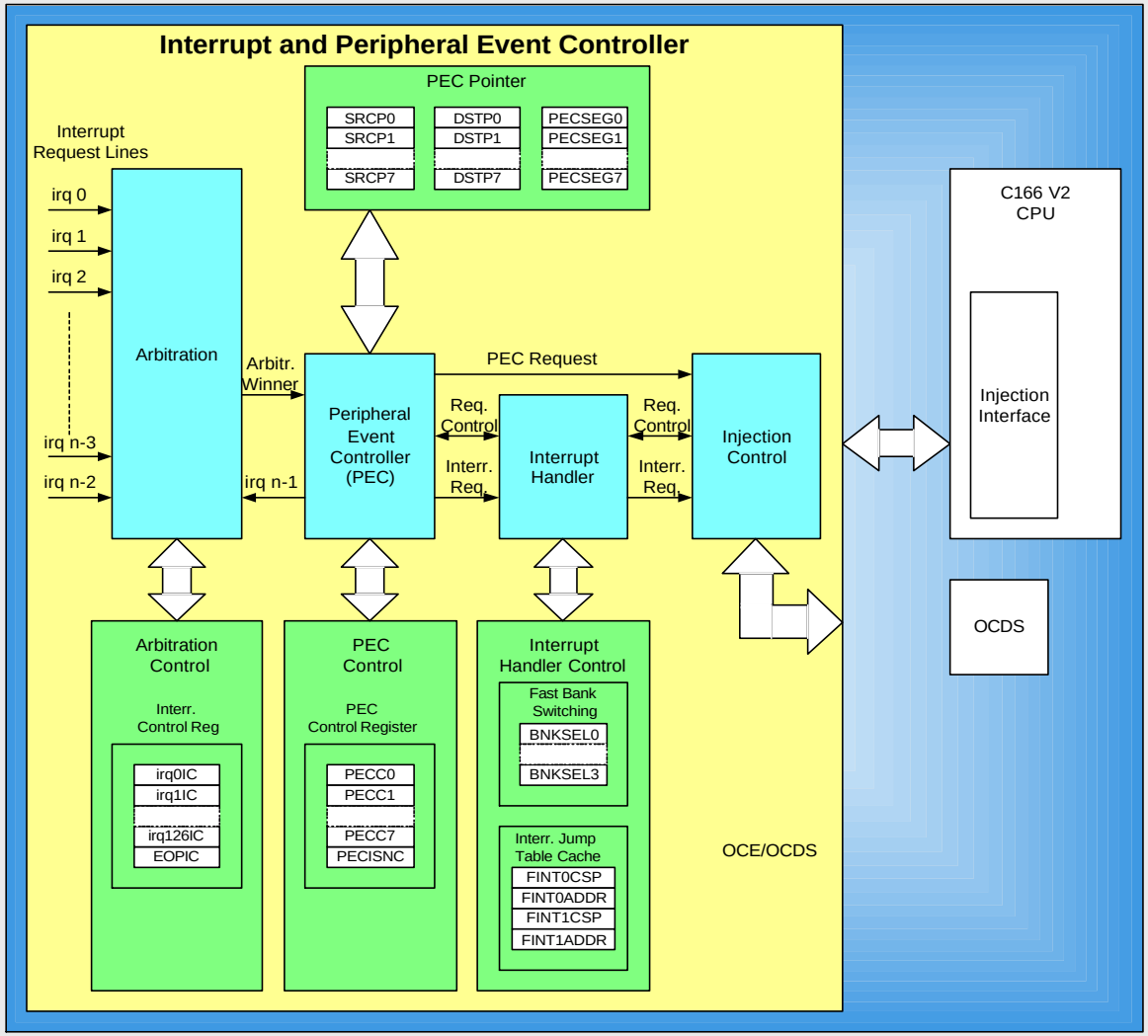
C166S-V2 CPU Architecture
Interrupt / Peripheral Event Controller

- Arbitration
  - 3 stage interrupt prioritization scheme, up to 128 interrupt nodes
- PEC Controller
  - up to 8 fast data(word/byte) transfers
- Interrupt Handler
  - fast bank switching, interrupt jump table cache
- Injection Control
  - handle interrupt. requests and PEC requests
- Control register and PEC pointer
  - interrupt Priority/group levels, source/destination, PEC channels

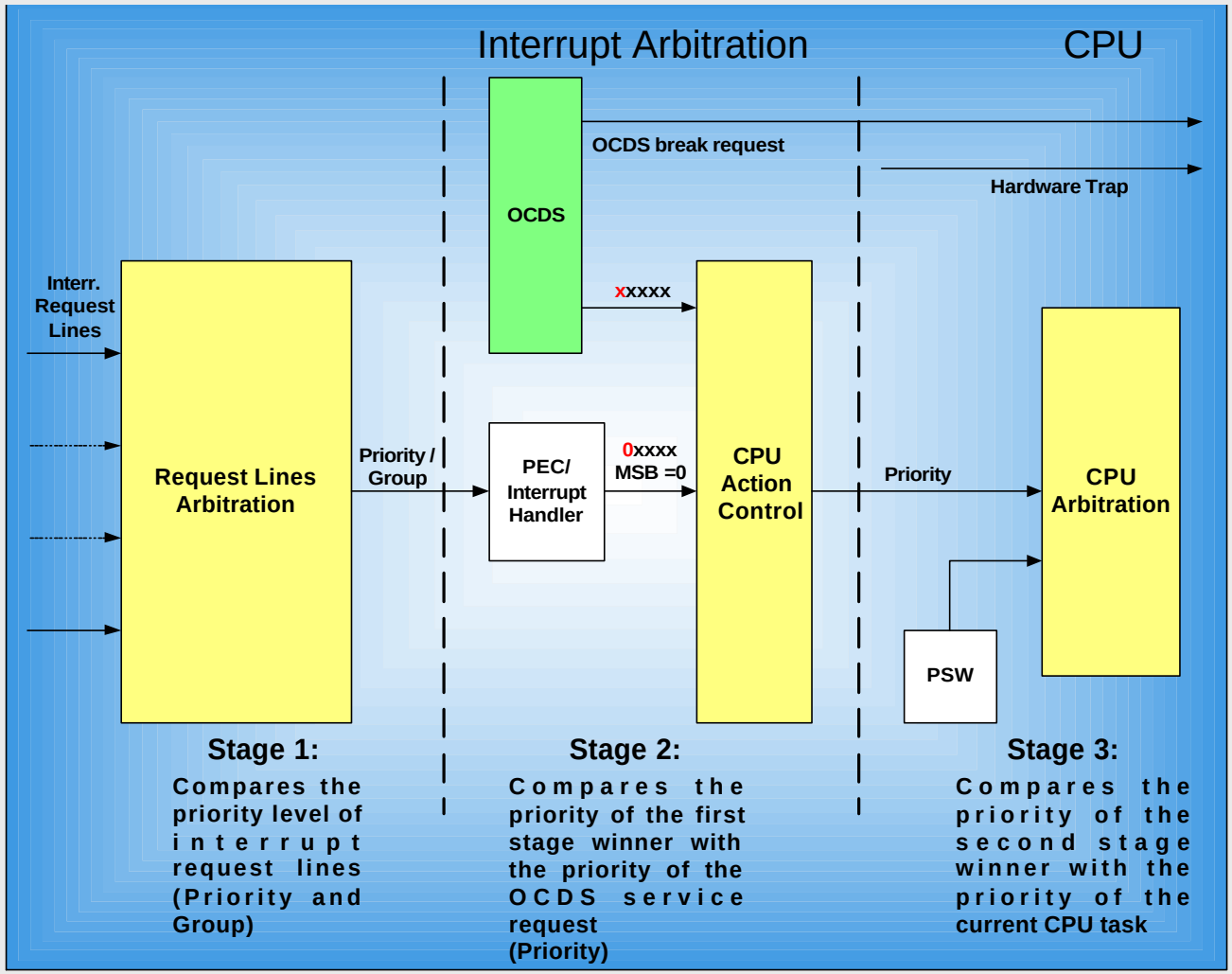
C166S-V2 CPU Architecture
Interrupt / Peripheral Event Controller



C166S-V2 CPU Architecture
Interrupt Arbitration

- The arbitration process starts with an enabled interrupt request and stays active as long as an interrupt request is pending:
  - The first arbitration stage compares the priority level
    - Priority level and group level
  - The second arbitration level compares the priority of the first stage winner with the priority of the OCDS service request
    - Each OCDS request with MSB = 1xxxx will always win the second stage arbitration
  - The third arbitration stage compares the priority of the second stage winner with the priority of the current CPU task
  - OCDS break requests and hardware traps bypass the arbitration scheme and go directly to the core.

C166S-V2 CPU Architecture
Interrupt Arbitration



C166S-V2 CPU Architecture
Peripheral Event Controller (PEC)

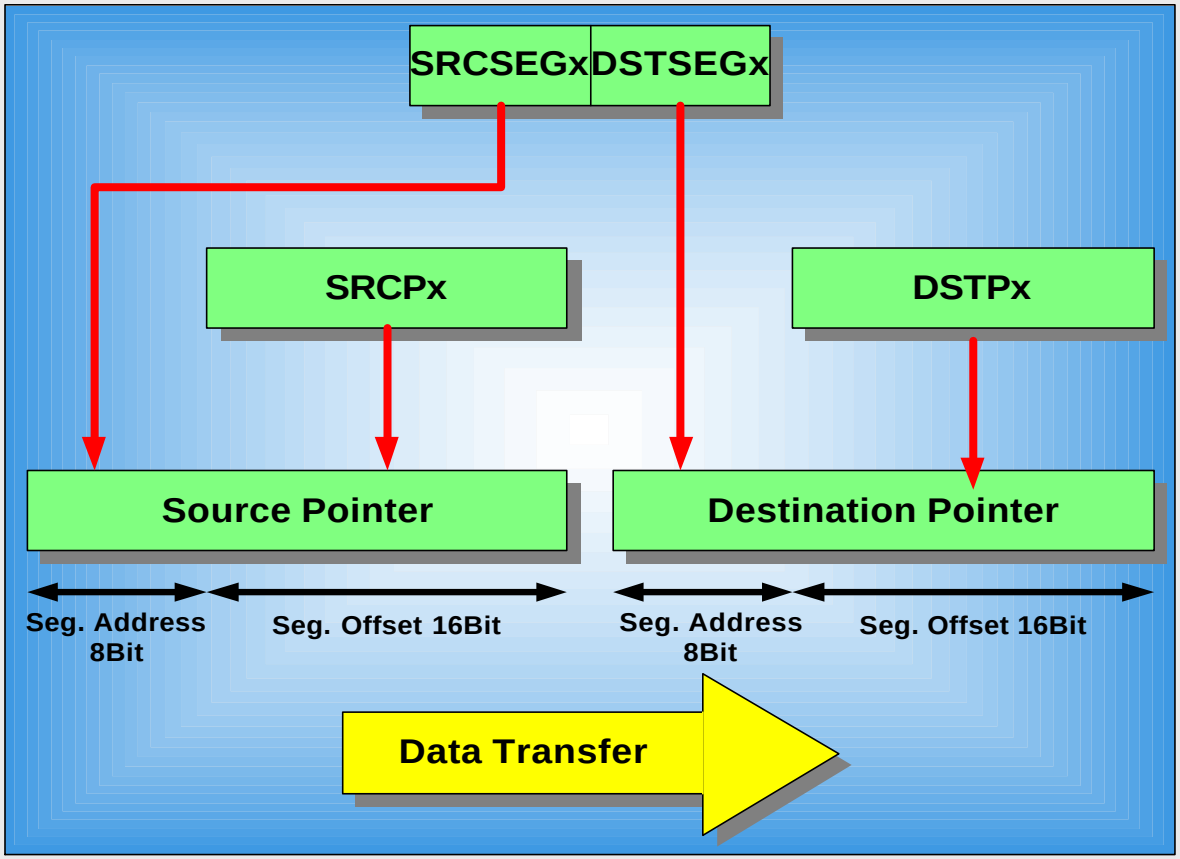
- PEC has been extended to allow both SRCPx and DSTPx to be modified automatically after a move
  - Allows PEC to move tables of data to / from anywhere in memory
- PECCx has an extra field to allow it to operate on levels 14,15, or 12,13, or 10,11, or 8,9 giving much more freedom to determine task priorities.
- An additional interrupt mode has been added for the 8 PEC channels to share. This is optionally triggered for End Of PEC (when the count goes from 1 to 0, and the data move is completed). This allows lower priorities to be set for the EOP which is typically not a time critical service.

C166S-V2 CPU Architecture
Peripheral Event Controller (PEC)

- EOPIC control has an additional sub-node indication register:
  - PECISNC holds which EOP channels have completed to trigger the EOP service routine
  - PECISNC is not cleared by interrupt entry, so must be handled by software

C166S-V2 CPU Architecture
Peripheral Event Controller (PEC)

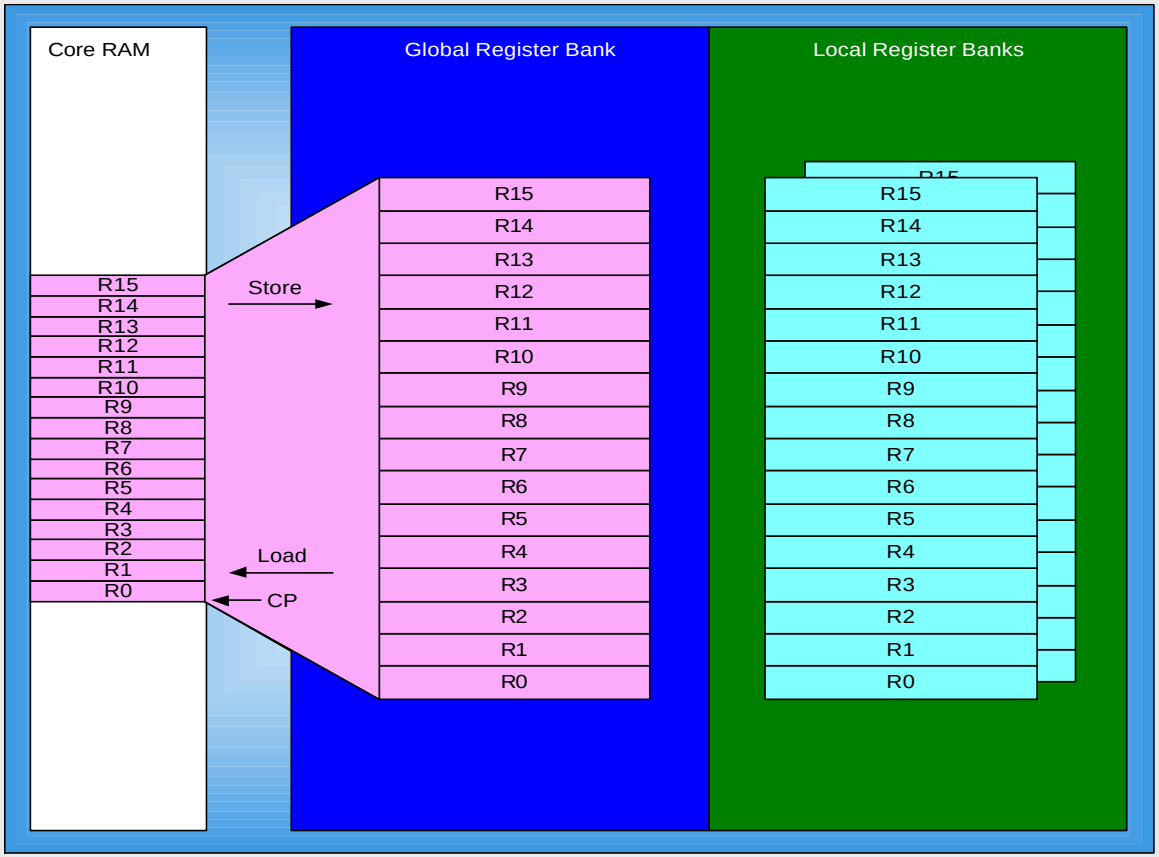
- PEC system is enhanced to have a segment address for each source and destination pointer.
  - Only SRCPx and DSTPx are modified automatically



XC166 architecture

C166S-V2 CPU Architecture
Register File

- After a context switch (e.g. function call,interrupt request) the contents of the former bank has been stored and the contents of the new bank has been loaded.



C166S-V2 CPU Architecture
Global Register Bank

- There is 1 global register bank with a 19 cycle context switch (950 ns @ 20 MHz)
  - Used for longer routines as context sensitive local parameter storage
  - Memory mapped, so can use normal user stack ([R0]).
  - Active register bank is shown by Context Pointer (CP) register
  - Updating CP starts a Store / Load sequence for the current / new GPRs
- After the CP has been updated a state machine starts to store and load the contents of the global register bank.
  - STORE contents of old register bank = 8 cycles
  - LOAD contents of new register bank = 8 cycles
  - Pipeline stall = 3 cycles to prevent any GPR reading until LOAD is done

C166S-V2 CPU Architecture
Global Register Bank

- Interrupting the Store / Load sequence is handled 2 ways
  - If the interrupt uses a local register bank, then interrupt is taken immediately, and Store / Load is cancelled
  - If the interrupt uses a global register bank, then interrupt is taken after the current Store or the Load phase is complete.
  - If the interrupt is in the Store phase, then Store is complete then the Load is cancelled.
    - If the interrupt is in the Load phase, then the load is completed.
    - If in the IRQ a new context switch is made then a new Store / Load is initiated.
    - On return, the outstanding Store / Load or Store is re-started
- Global register banks supported in Tasking by **_using** qualifier

Local Register Banks

- There are 2 local register banks with 0 cycle context switch
  - Both local register banks are not memory mapped
  - Should be used for time critical interrupt routines
  - Active register bank shown in PSW register
  - In case of an interrupt service, the bank switch is automatically executed by updating the PSW
  - Banks supported in Tasking by new procedure definition qualifiers:

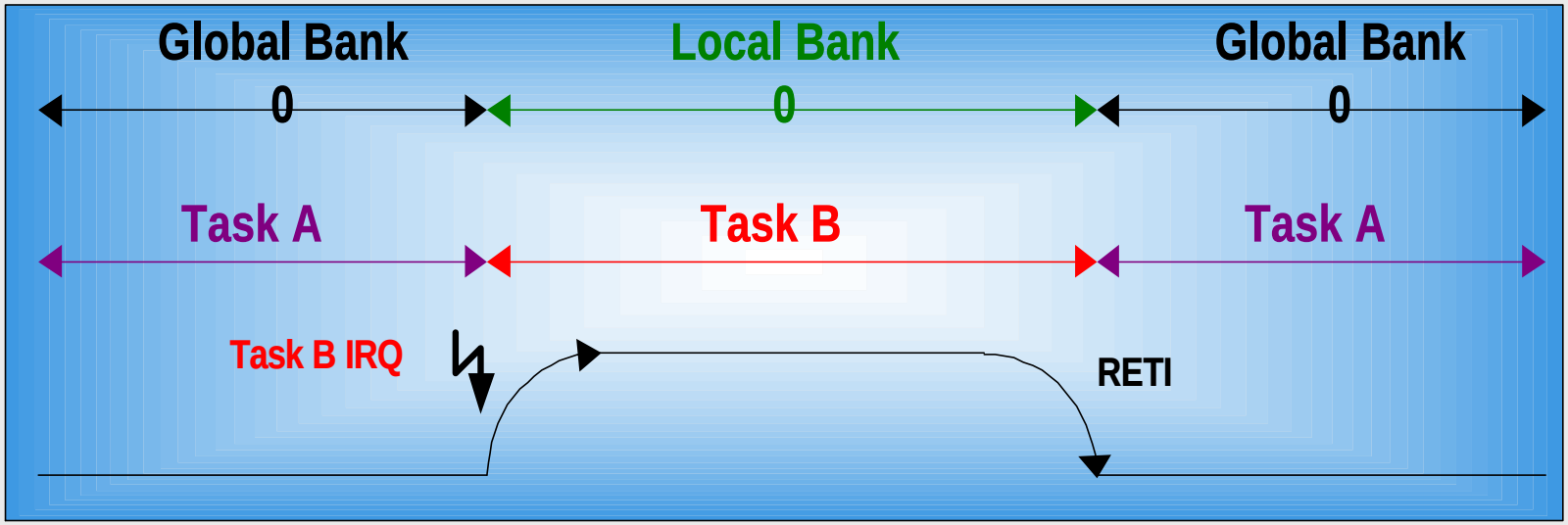
Local Register Banks

```
void _interrupt(0x10) _localbank(-1) _stacksize(+40) _cached
My_ISR(void)
{
        return;
}
```

- _localbank specifies 0,1,2. (0 = global bank)
- _stacksize specifies extra user stack provision for the new local user stack
- _cached uses FINT0ADDR and FINT1ADDR to bypass the interrupt vector table and directly inject the correct jump address of the interrupt service routine (to save time)

Context Switching (1)

- Context switch to local register bank
  - New bank is immediately available



Context Switching (2)

- Interrupted context switch to a new global register bank, interrupt task is combined with local bank
  - Local bank is available immediately
  - Interrupt occurs before store phase -> store / load again
  - Interrupt occurs after store phase -> load again