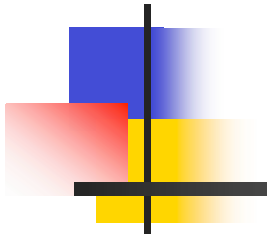
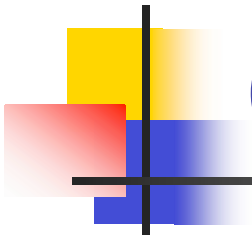


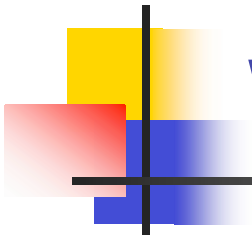
Device/driver Support of F3RP61

J. Odagiri



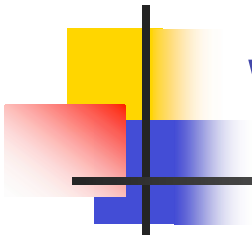
Contents

- What device/driver support does
- Implementation of device/driver support
- Accessing I/O modules
- Accessing sequence CPU devices
- Accessing shared memory
- Processing records by I/O interrupt
- Summary



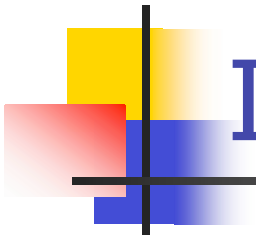
What devSup/drvSup does (1/2)

- FAQ
  - Which I/O modules does your device/driver support cover?
- The answer is "All", but…
  - The device/driver support modules support only basic access to the registers of I/O modules
  - That’s all if the I/O module is a simple one, such as DI, DO, A/D, D/A



What devSup/drvSup does (2/2)

- What if the I/O module is more complicated one, such as stepping motor controller?
  - Use EPICS sequencer (SNL) to implement control sequence
    - Wait for a directive to come in from OPI
    - Write a command code into a register
    - Set a command execution flag (Y-relay)
    - Wait for instantaneous ACK (X-relay)
    - Wait for FIN upon completion (X-relay)
    - Go back to the top

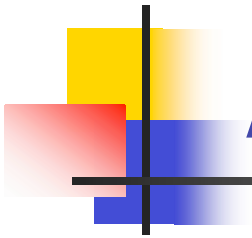


Implementation of devSup/drvSup

- Nothing special
- Just wrap the kernel-level driver
  - Everybody's doin' a brand new control now
  - Come on baby, do the F3RP61
  - I know you'll get to like it if you give it a chance now
  - My little baby sister can do it with ease
  - It's easier than learnin' your ABCs
  - So, come on, come on, and do The F3RP61 with me!
    - Ask a question about the origin of the above six lines to elderly people (such as Furukawa-san) if you are young enough not to know it

Accessing I/O modules (1/3)

- Digital I/O modules
  - Bit read/write
    - bi to read X and Y (relay status)
    - bo to write Y (relay status, X is not write-able)
  - Word read/write (16 bits)
    - mbbiDirect to read Xs and Ys (status bits)
    - mbboDirect to write Ys (operation mode etc.)
    - longin to read Xs and Ys (binary value)
    - longout to write Ys (binary value)
    - ai to read Xs and Ys (binary value)
    - ao to write Ys (binary value)



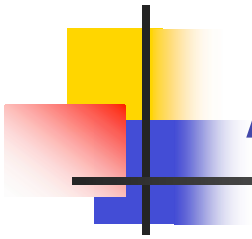
Accessing I/O modules (2/3)

- Analog I/O modules
  - longin, mbbiDirect
    - to read back parameter registers
  - longout, mbboDirect
    - to write parameter registers
  - ai
    - to read input/output data registers
  - ao
    - to write output data registers

Accessing I/O modules (3/3)

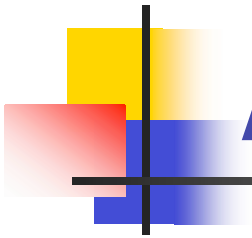
- An example
  - Read X2 of DI in slot 3
  - All modules are on unit 0

```
record ( bi, "test:slot3:ch2" )
{
        field ( SCAN, "1 second" )
        field ( DTYP, "F3RP61" )
        field ( INP, "@U0,S3,X2" )
}
```



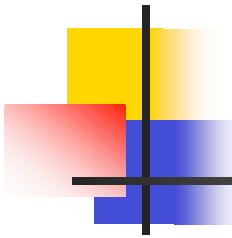
Accessing sequence CPU

- Data register (D), Internal relay (I), etc.
- Needs message passing to access those devices
  - Send a command message and wait for the response message to come in
  - Must implement asynchronous I/O device support
- Under construction (not yet finished)
- More efficient inter-CPU communication has been supported based on shared memory



Accessing shared memory (1/3)

- One unit of FA-M3 can have up to 4 CPUs (sequence CPUs and F3RP61s)
- They can communicate each other over the shared memory
- Typical application that requires shared-memory-based communication is monitor of interlock system
  - Sequence CPUs handle interlock logic and copy the I/O-relay status into shared memory
  - F3RP61 reads the status from the shared memory and report to the CA-clients



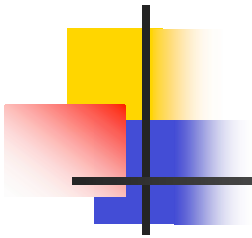
Accessing shared memory (2/3)

- An important caution (from our experience)
  - Never read I/O-relays by using F3RP61 directly
  - Otherwise, rebooting Linux on F3RP61 can make the ladder programs stop by I/O-Error
  - Use shared memory to transfer the I/O status read by the sequence CPUs to F3RP61

Accessing shared memory (3/3)

- An example
  - Read one word of data from address 0 in a shared memory area owned by a sequence CPU in slot 1
  - All modules are on unit 0

```
record ( mbbiDirect, "test:cpu1:addr0" )
{
        field ( SCAN, "1 second" )
        field ( DTYP, "F3RP61" )
        field ( INP, "@CPU1,R0" )
}
```



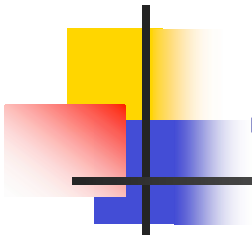
Processing by I/O-interrupt (1/2)

- Digital input module can cause I/O-interrupt
- Kernel-level driver of F3RP61 transforms the interrupt into a message and delivers it to a user process
- Device/driver support supports the scan value of "I/O Intr" based on the kernel-level driver
- Most of input-type records can use "I/O Intr"
- Under construction (Not yet finished)

Processing by I/O-interrupt (2/2)

- An example
  - Read CH2 of A/D module in slot 5 upon an interrupt on X1 of DI in slot 2
  - All modules are on unit 0

```
record ( ai, "test:read_adc:upon_intr" )
{
        field ( SCAN, "I/O Intr" )
        field ( DTYP, "F3RP61" )
        field ( INP, "@U0,S5,A2:U0,S2,X1" )
}
```



Summary

- You can access all types of the I/O modules
- Use EPICS sequencer (SNL program) to implement control sequence if necessary
- Use shared memory in order to communicate with sequence CPUs
- Message-based access to CPU devices (D, I, etc.) will be supported sometime in the future
- I/O-Interrupt will be supported (very) soon