APVDAQ Software

Takeo Higuchi (Kavli IPMU (WPI))

Disclaimer …

- **The APVDAQ-software R&D has to be suspended for ~3 weeks due to a higher priority job. I have been concentrated on the DAQ-software R&D as an ex. member of the DAQ in order for the telescope test.**
  - We (DAQ + TH) are here to establish data streams from ONSEN to EVB (main data stream) and from HLT to ONSEN (RoI data stream), and to establish the RoI-based hit rejection scheme.
  - The connectivity test will finish by the end of my stay in DESY at the longest. I will return to the APVDAQ-software R&D immediately after that.

Mission Reminder

- **Conversion of a Windows-version readout software of APVDAQ to Linux version.**
  - Windows-version software (GUI) … already there.
  - **Linux-version software (GUI) … needs developed by TH.**
    - **I prefer CUI instead of GUI ← the initial version will be CUI based.**
- **Conversion of the VME-controller API of National-Instruments to CAEN.**
  - Windows-version software uses NI VME controller.
  - **Linux-version software uses CAEN VME controller (V2718).**

Software Development Policies

- **Hitchhiking the Windows**
  - **Fast and easiest way.**
  - **Hard to understand the APVDAQ-controlling raw protocol → any tiny problem, demands Markus-san to solve.**
  - **Hard to understand the software structure, as well → GUI may be developed.**
- **Reinvention the wheel**
  - **Slow and difficult way.**
  - **Able to understand the APVDAQ-controlling raw protocol → software can flexibly be developed.**
  - **Attractive way for an ex.DAQ group member.**

Yes, I know the ladder production phase is approaching very soon, but let me take "reinvention the wheel" way for better APVDAQ understanding.

Abstraction [1]

- **Abstraction of VME access with CAEN V2718**
  - The CAEN VME library hides VME access primitives, which is a bit annoying implementation.
    - I like: fd = open("/dev/vmea32d32", O_RDWR);
      mmap(start, length, PROT_READ|PROT_WRITE, MAP_SHARED, 0);
  - Anyway, I managed to **abstract VME R/W of 1, 2, and 4 byte data** with V2718, **except DMA access**.
    - If there exist VME interrupt register, it would be very helpful for R&D and debug...

Abstraction [2]

- **APVDAQ register list**

Looks much more complicated than the COPPER registers … to me

Group	BASE	NC	D	GROUP	ADDR	NC	Hex	Direction	Name	Type	Data				Data Width	
	A[31..24]	A[23..16]	A[15]	A[14..8]	A[7..1]	A[0]					D[3]	D[2]	D[1]	D[0]	Bits	Bytes
APV-I2C	10111011	00000000	0	0000000	0000000	0	0xbb000000	Read	READ_AI2C_WORD		D[7..0]: data word				8	2
	10111011	00000000	0	0000000	0000010	0	0xbb000004	Read	READ_AI2C_ERR		lbyte not ack'd	wbyte not ack'd	addr not ack'd	bus busy	4	2
	10111011	00000000	0	0000000	0000000	0	0xbb000000	Write	WRITE_AI2C_WORD		D[7..0]: data word 0, D[15..8]: data word 1, D[23..16]: data word 2, D[31..24]: data word 3				32	4
	10111011	00000000	0	0000000	0000010	0	0xbb000004	Write	WRITE_AI2C_ADDR_RW		D[9..8]: # of words, D[7..1] address			0=write; 1=read	10	2
	10111011	00000000	0	0000000	0000100	0	0xbb000008	Write	WRITE_AI2C_RESET	static			I2C reset		2	2
TLC	10111011	00000000	0	0000000	0100000	0	0xbb000040	Read	READ_TLC					D-OUT	1	2
	10111011	00000000	0	0000000	0100000	0	0xbb000040	Write	WRITE_TLC	static		CS	AD-INP	IO-CLK	3	2
PHOS-I2C	10111011	00000000	0	0000000	0010000	0	0xbb000020	Read	READ_PI2C_WORD		D[7..0]: data word				8	2
	10111011	00000000	0	0000000	0010010	0	0xbb000024	Read	READ_PI2C_ERR		lbyte not ack'd	wbyte not ack'd	addr not ack'd	bus busy	4	2
	10111011	00000000	0	0000000	0010000	0	0xbb000020	Write	WRITE_PI2C_WORD		D[7..0]: data word 0, D[15..8]: data word 1, D[23..16]: data word 2, D[31..24]: data word 3				32	4
	10111011	00000000	0	0000000	0010010	0	0xbb000024	Write	WRITE_PI2C_ADDR_RW		D[9..8]: # of words, D[7..1] address			0=write; 1=read	10	2
	10111011	00000000	0	0000000	0010100	0	0xbb000028	Write	WRITE_PI2C_RESET	static			I2C reset		2	2
JTAG	10111011	00000000	0	0000000	0001100	0	0xbb000018	Read	READ_JTAG		D7: TDO				8	2
	10111011	00000000	0	0000000	0001100	0	0xbb000018	Write	WRITE_JTAG	static	D6: TDI		TMS	TCK	7	2
NCONFIG	10111011	00000000	0	0000000	0110000	0	0xbb000060	Write	WRITE_NCONFIG	static				START_LD_ALT1	1	2
LRES	10111011	00000000	0	0000000	0111110	0	0xbb00007c	Write	WRITE_LRES	pulse	D31: reset				32	4
CONF	10111011	00000000	1	1010000	0000000	0	0xbb00d000	Write	READ_CONF		D8: RES0, D7: PWR0, D6: PHOS4_PWR, D5: 0=SR; 1=Seq; D4: ADC_S2				9	2
	"	"	"	"	"	"	"	"	"		ADC_S1	FIFO CLK polarity	Enable RESERV	Enable internal tick/frame detection		
	10111011	00000000	1	1010000	0000000	0	0xbb00d000	Write	WRITE_CONF	static	same as READ_CONF				9	2
	10111011	00000000	1	0101000	0000000	0	0xbb00a800	Write	WRITE_CNT	static	D[27..24]: allowed trgs, D[23..16]: SR Delay, D[15..7]: Frame length, D[5..0]: Tick length				28	4
TRG	10111011	00000000	1	0011000	0000000	0	0xbb009800	Write	WRITE_TRG	pulse					0	2
	10111011	00000000	1	0100000	0000000	0	0xbb00a000	Read	READ_VETO					veto status	1	2
	10111011	00000000	1	0100000	0000000	0	0xbb00a000	Write	WRITE_VETO	static (indir.)				veto status	1	2
	10111011	00000000	1	0100100	0000000	0	0xbb00a400	Write	READ_EVENTNR		D[31..0]: event counter from TLU - ONLY APPLICABLE WITH TLU-ADAPTER INSTALLED				32	4
SRSeq	10111011	00000000	1	0110000	0000000	0	0xbb00b000	Write	WRITE_TRG_SEQ	pulse train	RAMDAC	SoftwareTrigger	CalRequest	SoftReset	4	2
	10111011	00000000	1	1000000	0000000	0	0xbb00c000	Write	WRITE_SEQ_RAM	A[9..2]=addr	HardwareTrigger	SoftwareTrigger	CalRequest	SoftReset	4	2
	10111011	00000000	1	0111000	0000000	0	0xbb00b800	Write	WRITE_DAC_RAM	A[9..2]=addr	D[29..20]: B (User), D[19..10]: G (Repeater), D[9..0]: R (ADC)				30	4
FIFO	10111011	00000000	1	0001000	0000000	0	0xbb008800	Read	READ_FIFO_AB		D[9..0]: ADC A, D[12..10]: Tick, Frame, AUX_OUT0, D[25..16]: ADC B				26	4
	10111011	00000000	1	0010000	0000000	0	0xbb009000	Read	READ_FIFO_CD		D[9..0]: ADC C, D[25..16]: ADC D				26	4
TESTPINS	10111011	00000000	1	1111100	0000000	0	0xbb00fc00	Write	WRITE_TP_SEL	static			Sel1	Sel0	2	2

Abstraction [3]

- **Register access of APVDAQ**
  - **APV-I2C … communication with APV25**
  - **PHOS-I2C … control of PHOS4 4ch-delay ASIC for trigger delay**
  - **NCONFIG … ALTERA download register**
  - **LRES … clear internal FIFO of APVDAQ (static)**
  - **CONF-REG … APVDAQ configuration (8 sub configs)**
  - **CNT-REG … counters (4 sub counters) ← ☹☹**
  - **TEST-TRG … test trigger generation ← ☹**
  - **VETO … veto of triggers ← ☹**

Abstraction [4]

- **Register access of APVDAQ – cont'd**
  - **SEQ-HW-TRG … control of HW trigger buffer (?) ← ☹☹☹**
  - **SEQ-SW-TRG … control of x3 SW trigger buffer (?) ← ☹☹☹**
  - **SEQ-RAMDAC … control of x3 10-bit DAC buffer (?) ← ☹☹☹**
  - **SEQ-SINGLE-PEAK-MODE … ? ← ☹☹☹☹☹**
  - **SEQ-MULTIPLE-PEAK-MODE … ? ← ☹☹☹☹☹**
  - **FIFO … direct R/O from the ALTERA FIFO (?)**
  - **TEST-PINS … output pattern on test pins**

Abstraction [5]

- **Abstraction:**
  - I find access to APV and PHOS4 via I2C bus is specially complicated than other registers → I like to hide this complication from users.
  - In some cases, one register (32 bits) has multiple meaning (it looks like so, to me) → I like to deconvolve the register to each meaning.
  - My preference/grand goal is “ioctl” like abstraction as:
    apvdaq_ctl(module, APVDAQ_PHOS4_READ);
    apvdaq_ctl(module, APVDAQ_VETO_SET, 0);
    - **I fully agree that my abstraction desire is be too much in this critical phase of the short-coming ladder assembly.**

Abstraction [6]

- **What I like-to/should know at the *minimum* level.**
  - **How to make cold-start of APVDAQ.**
  - **How to start the data taking. How to stop the data taking.**
  - **How to readout the stored data in FIFO (no DMA needed).**
  - **How to select trigger source, and in case of the APVDAQ-generated trigger, how to preset trigger rate and # of triggers.**
  - **And anything important I'm missing.**
    - In case of any accidents out of the scope above, I just reset the APVDAQ boards.
  - **I strongly like Markus-san to very kindly spare his time to teach me the issues listed above.**
    - I have APVDAQ-control software (under R&D by me) in a DESY computer.