WEP4P2

New EPICS Channel Access Backend

Akio Morita(SAD committer in SuperKEKB commissioning group),
KEK, 1-1, Oho, Tsukuba, Ibaraki 305-0801, Japan

Motivation

SADScript has 2-kind of EPICS CA interfaces. One is synchronous CaRead/CaWrite function interface based on ca_get & ca_pend_io API. The other is asynchronous CaMonitor class interface based on ca_add_masked_arrat_event & ca_pend_event API. These code are written in old EPICS CA library APIs and contain many magic numbers. There have following issues for current SuperKEKB operation and code maintenece.

- Different(inconsistent) data transder codes for CaRead & CaMonitor.
- ca_pend_io invoked by CaRead prevents CA event handling for CaMonitor.
- CaRead causes IOC side MAX_ARRAY_BYTE limit for waveform PV due to DOUBLE transfer.
- CaRead causes error on ENUM PV holding value without symbol-definition due to STRING transfer.
  For example:
  IOC in KEK PF ring has such ENUM PV value.
- CaMonitor can't control # of elements to transfer.
  For exsample:
  CaMonitor for SuperKEKB needs 100Mbps/TbT-BPM.
  Core switch fabric: 10Gbps / Server interface: 1Gbps
- CaRead for multiple PVs is inefficient due to ca_get ca_pend_io loop.
  It has posibility to offer multiple ca_get for low letency PVs and causes different semantics if GET acction has side effect in IOC implement.

CaRead Issue

* Original(Old) CaRead is implemented by ca_get & ca_pend_io.
 - ca_pend_io blocks until timeout or ca_get request completion.
 - ca_pend_io does not process CA background tasks.
 - CaRead against PV with long access latency delays CaMonitor callback and will drop callback events if event rate is enough high to overflow receive buffer.
 - This issue discourages to use both CaMonitor and CaRead in combination, however, CaMonitor class does not support synchronous get action.

* CaRead supports PV name argument by implicit CaOpen which is implemented by ca_search & ca_pend_io.
 - CaRead named access to PV on downed IOC MIGHT be blocked during 2.2sec which is CaOpen timeout in original implementation. This delay would make serious side effect for CaMonitor callback processing.

* CaRead supports concurent GET for multiple Pvs.
 - It is implemented by multiple ca_get requests for Pvs and single ca_pend_io for waiting ca_get completion.
 - If ca_pend_io is timed-out, CaRead replays ca_get request for all Pvs including succeeded PV at previous ca_get action and repeat ca_get/ca_pend_io until full success or maximum iteration.
 - This ca_get replay algorithm makes multiple GET requests for same PV if CaRead arguments contain long latency IOC's PV.
If GET action has side effect(for example, PV returns number of accesses), concurrent CaRead result becomes non-deterministic. But, replacement of concurrent CaRead with sequential chain of CaReads makes slow code due to network round trip time.

Reimplementation Concepts

- Implement backend bridge codes based on EPICS 3.15 CA library APIs.
- Unified data packing & unpacking code for both CaRead/CaWriter and CaMonitor.
- Introduce event callback based CaRead backend by using ca_get_array_callback with transfer control memorandum for detecting completion or canceling callback.
- Transfer data by native data types if possible.
- Introduce transfer data type & length customize interfaces.
- Introduce timeout configuration interfaces.
- Implement backward compatible user interface SIM.
- Improve code readability & portability.
- Improve debug information.

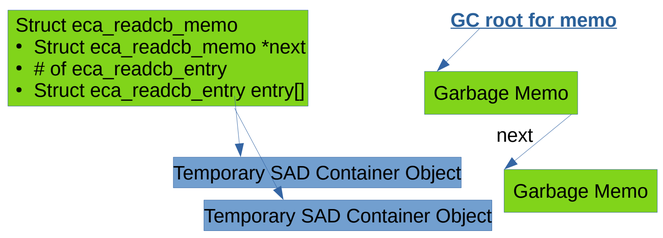
New Features

- Configurable CaOpen & CaRead timeout.
- Data trasfer type/length modifier for PV descriptors on CaRead/CaWrite interfaces.
- Parallel ca_get/ca_put for the nestted PV descriptor lists on CaRead/CaWrite interfaces.
- Data transfer type/length configuration option for CaMonitor class.
- WaitValue class method to block multiple CaMonitor instances until either all value filled by CA ValueCB or timed-out.
- Lazy ca_flush_io support for Put instance method on CaMonitor class for improving network efficiency in multiple Puts.
- New code CAN compile as dynamic loadable extension module.

Note: DbStatic.c needs new builtin function table extension(not back-ported at this moment).

GET Callback Memorandum

~ workaround for cancelling ca_get_array_callback request ~



Garbage Collection
1. Scan single linked list pointed by "GC root for memo".
2. If "entry"s in eca_readcb_memo are already completed, unlink memo from single linked list and deallocate it. (Lazy deallocation of cancelled memo)
3. Process GC until end of linked list.

Callback routine
1. Receive eca_readcb_entry via given user pointer.
2. Check SAD object referance in eca_readcb_entry.
3. Store result into referenced SAD container object if valid reference.
4. Mark eca_readcb_entry as completed.

CaRead backend
1. Allocate eca_readcb_memo and temporary SAD container to return result.
2. Set container reference into eca_readcb_entry.
3. Set timeout counter.
4. Offer ca_get_array_callback with eca_readdb_entry pointer.
5. ca_pend_event(time-slice); timeout -= time-slice
6. Test ca_get completion by scanning eca_readcb_entry.
7. return (5) if timeout > 0 && uncompelted callback exists.
8. Deallocate eca_read_memo & Return SAD container object If all callbacks have been completed. (success without timeout)
9. Timed-out case:
- Fill uncompleted part of SAD container by $Failed.
- Invalidate uncompleted part of eca_reacb_entry (cancel writeback).
- Insert eca_readcb_memo to single linked list pointed by "GC root for memo".
- Return SAD conrainer object.

Code Implementations

tfEPICSCA_.c

- Resorce manager for chid & evid.
- Resorce manager for get callback memorandum.
- Callback routine for CaRead.
- SADScript frontend functions for EPICS Channel Access operations.
- SADScript frontend functions to configure timeout and debug level.

epicsca_callback.f

- Callback routines for ConStatCB, ValueCB, PutCB to write-back values into CaMonitor instance.

DbStatic.c

- Wrapper for dbStaticLib.h

CAManager.n

- CaRead/CaWrite interface wrapper
- CA connection pool for CaRead/CaWrite

CaMonitor.n

- CaMonitor class interface wrapper

Major User Interface

EPICS CA backend configuration interfaces:
* EPICS$CaDebugPrint[level_Real]
 Customize CA backend debug log level. (default **0**)
* EPICS$CaOpenTimeout[timeout_Real]
* EPICS$CaReadTimeout[timeout_Real]
 Customize ca_create_channel / ca_get timeout. (default **10sec**)

CaRead/CaWrite synchronous interfaces:
* CaWrite[General-PV-descriptor, value] – Send value by ca_put & CaRead[]
* CaRead[General-PV-descriptor]
 – Offer ca_get & Returns {value, status, severity, timestamp} or $Failed(timeout).
 Note: Epoch of "timestamp" is EPICS epoch. (It is not either UNIX time or SAD time.)

* CaRead[pv-list_List] ~ Map[CaRead, pv-list]
 Note: ca_gets for pv-list execute concurrently.

* CaWrite[pv-list_List, val-list_List] ~ MapThread[CaWrite, {pv-list, val-list}]
* CaWrite[pv-val-list_List] ~ Apply[CaWrite, pv-val-list, {1}]
 Note: ca_puts & ca_gets for pv-list execute concurrently.

Syntax of PV descriptor for CaRead/CaWrite interfaces:
```
General-PV-descriptor := Typed-PV-descriptor
                       | Rule[Typed-PV-descriptor, transfer-length_Real]
Typed-PV-descriptor   := PV-descriptor
                       | String[PV-descriptor] (offer STRING transfer)
PV-descriptor         := chid_Real | PV-name_String
```

- Default transfer-length is ca_element_count().
- "0" transfer-length means automatic length by IOC same as CaMonitor default.

CaMonitor class acynchronous interface:
* Constructor options
 - AutoStart – ca_create_subscription at connection establishment (default **True**)
 - ValueType – Transfer data type (default **Automatic**)
 - Count – Transfer element count (default **0** *decided by IOC*)
 - Flush – ca_flush_io timming (default **True**)
  True: Immediate ca_flush_io / **Lazy**: Lazy ca_flush_io / **False**: Never ca_flush_io
 - ConStatCommand – Callback at connection state change
 - ValueCommand – Callback at PV change
 - PutCommand – Callback at PutCB[] completion

* Instance Methods
 - ConnectedQ[] - Return True if PV is connected.
 - ConStat[] - Return EPICS connection state code.
 - Start[] - Start monitoring by ca_create_subscription.
 - Stop[] - Stop monitoring by ca_clear_subscription.
 - FlushIO[] - Explicit ca_flush_io.
 - Value[] - Return value of last value callback event (including GetCB[]).
  Note: Value[] return **Undefined** before first value callback.
 - TimeStamp[] - Return SAD timestamp of last value callback event.
  Note: TimeStamp$[] returns EPICS timestamp same as CaRead[].
 - Severity[] - Return severity of last value callback event.
 - Put[val_] - Offer ca_put "val" and invoke ca_flush_io if required by Flush option.
 - PutCB[val_] - ca_put_callback variation of Put[]
 - GetCB[] - Offer ca_get_callback. (This method CAN update Value[] without monitoring)
 - WaitValue[timeout_Real] – Wait Value[] filling until timeout and return boolean if Value[] is filled.

* Class Method
 - CaMonitor@WaitValue[timeout_Real] – Concurrent WaitValue[] for all CaMonitor instance.