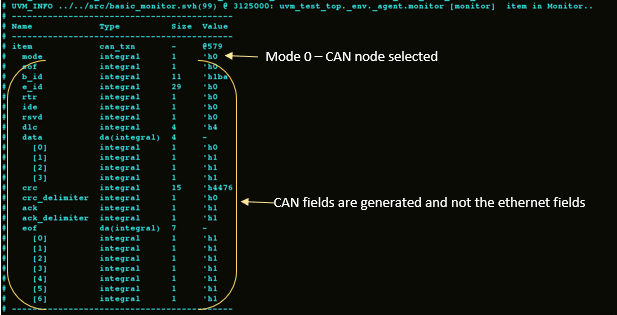
**Override UVC:**

The code is bit messy in variable UVC approach, if the UVC supports a greater number of protocols as all the protocols needs to be coded in a single file. It makes the process more complicated, if there occurs an error that needs to be debugged and then it becomes difficult to decide on which protocol the error is. There is a method in UVM called overrides which is useful in overriding the class only the class is registered in factory. The overriding is done in basic test file. Two transaction needs to be created: one for CAN and other one for ethernet. Both the transactions need to be extended from the basic transaction. The basic transaction has some common variables for the protocol used. Mode and the inter\_frame\_gap are the common variables, so they are declared in the basic transaction. The can variables are declared in the CAN transaction, similarly for ethernet. If the mode is ‘0’ then the basic transaction is overridden by CAN transaction by set\_type\_override\_by\_type method. The base class is the first argument and the child class object is the last argument for that method. If there is an error in the CAN pack/unpack methods, then the debug is much simpler compared to the multiple UVC. Since, the file contains only one protocol logic. Observe the log files in Figure 10 and 11, now the transaction has only CAN variable because the mode value is ‘0’.

Figure 10: CAN transaction from monitor with mode 0

Mode 0 is selected from the test component. Variable UVC’s sequence item has both the CAN members and the ethernet members, but only basic sequence item is present for override UVC approach. The appropriate transaction items override the basic sequence item based on the mode value. CAN transaction item file is selected, because mode variable carries the value 0. So, CAN members only present in both the interface and the sequence item. Now the monitor captures only the CAN member values and it does not bother about the ethernet members which is shown in Figure 10.

Unlike variable UVC there are no overhead present in driver to segregate the signals based on the mode value. Driver gets the randomized variables from the sequence and directly transmits the same in to the bus. Randomized CAN members are received in the driver because of the mode value 0 and there are no ethernet members. Now the CAN transactions are transmitted serially in the CAN bus which is shown in Figure 11. Driver logic is independent of mode variable. Thus, UVC will not get disturbed if another protocol got added. Only change required is that the new protocol needs to override the basic transaction item. Variable UVC changes mostly all the components of UVC which consumes more time, but the override UVC method adds only a single line to the existing code which is an efficient and a faster way.

A screenshot of a cell phone

Description automatically generated

Figure 11: CAN transaction from driver with mode 0

After changing the mode variable to ‘1’ then the UVC acts as an ethernet node and the results are shown in Figure 12 and 13. Here, only the ethernet members are present in the transaction of the override UVC which is shown in Figure 12. The driver converts the UVC’s transaction into ethernet signals and transmits it over the bus.

Monitor does an exact opposite function of driver. The transmitted values are broadcasted into the UVC bus. The signals got sampled from the interface once the proper control signals are received. Then the collected samples are converted into the transaction of override UVC type. Ethernet frames are decoded into the respective ethernet members via the unpack system function present in the uvm\_transaction base class. The monitor transaction is shown in Figure 13.



Figure 12: Ethernet transaction from monitor with mode 1

A screenshot of a computer

Description automatically generated

Figure 13: Ethernet transaction from driver with mode 1