# C7x queries

## 1. Vector pointing to a vector

```
//An array of 10 vectors of type int16
int16 vData[10];

//Initialize vData
for(int32_t idx = 0; idx < 10; idx++) {
    vData[idx] = (int16)(idx * 1000);
}
```

```
//A vector pointer of type int16 pointing to 3rd vector of vData
int16 *pData = (int16 *)&vData[2];

//Read the vector at pData
int16 vD = *pData;
```

The above declaration of vector pointing to an array of vectors from the “C7x vectors tutorials(Chapter 2,Basics 2)” gives the below error -

```
test1.cpp: In function ‘int main()’:
test1.cpp:89:26: error: invalid cast from type ‘_c70_he_detail::vtype_ptr<int, 1
6>’ to type ‘int16* {aka _c70_he_detail::vtype<int, 16>*}’
 pData = (int16 *)&vData[0];
                          ^
```

Similar error is also observed while running the “ FIR filter using Streaming Engine tutorial(Chapter 2, Sample 3)” -

**Declration -**

```
seTemplate = __SE_SET_PARAM_3D(&seParam, &seFlags);

*stov_ptr(__STRM_TEMPLATE, (uint8_t *)&pBlock[SE0_PARAM_OFFSET]) = seTemplate;
```

**Error -**

```
test7.cpp: In function ‘int main()’:
/home/kpit/psdk_rtos_auto_j7_06_01_00_15/ti-cgt-c7000_1.2.0.STS/host_emulation/include/vector.h:7975
:48: error: no matching function for call to ‘_c70_he_detail::vtype_ptr<unsigned int, 16>::vtype_ptr
(uint8_t*)’
 #define stov_ptr(type, addr) ((type##_ptr)(addr))
                                                ^
test7.cpp:69:2: note: in expansion of macro ‘stov_ptr’
 *stov_ptr(__STRM_TEMPLATE, (uint8_t *)&pBlock[SE0_PARAM_OFFSET]) = seTemplate;
  ^~~~~~~~
```

## 2. Can SE_flags and SE_parms be passed to SE_template for SE_1 after __SE0_OPEN has been passed ?

In “FIR filter using SE tutorial(Chapter 2, Sample 3)”, why is SE_template for SE_0 and SE_1 stored in buffer pblock[128] before passing it to __SE0_OPEN and __SE1_OPEN ?

## Code -

```
seTemplate = __SE_SET_PARAM_3D(&seParam, &seFlags);

*stov_ptr(__STRM_TEMPLATE, (uint8_t *)&pBlock[SE0_PARAM_OFFSET]) = seTemplate;

//Setup Streaming Engine 1 to fetch and duplicate coefficients
seFlags = __SE_FLAGS_default();

seFlags.ELETYPE   = __SE_ELETYPE_16BIT;
seFlags.ELDUP     = __SE_ELEDUP_32X;
seFlags.VECLEN    = __SE_VECLEN_32ELEMS;

seParam.ICNT0 = COEFF_SIZE;
seParam.ICNT1 = INPUT_SIZE/SIMD_WIDTH;  seParam.DIM1  = 0;
seParam.ICNT2 = 1;                      seParam.DIM2  = 0;

seTemplate = __SE_SET_PARAM_3D(&seParam, &seFlags);

*stov_ptr(__STRM_TEMPLATE, (uint8_t *)&pBlock[SE1_PARAM_OFFSET]) = seTemplate;
```

```
int16_t *pIn    = &inArray[0];    //pointer to input array
int16_t *pCoeff = &coeffArray[0]; //pointer to coeff array
int16_t *pOut   = &outArray[0];   //pointer to output array

int32_t itr = 0;

//Read template and open Streaming Engine 0
seTemplate = *stov_ptr(__STRM_TEMPLATE, (uint8_t *)&pBlock[SE0_PARAM_OFFSET]);
__SE0_OPEN(pIn, seTemplate);

//Read template and open Streaming Engine 1
seTemplate = *stov_ptr(__STRM_TEMPLATE, (uint8_t *)&pBlock[SE1_PARAM_OFFSET]);
__SE1_OPEN(pCoeff, seTemplate);
```

### 3. Streaming address generator multi dimension stores not possible

In “SA multi-dimension stores example(Chapter 3,SA Example 2)”, error is thrown while storing a vector in image buffer.

**Code -**

```
//Fetch input data
for(int32_t ctr = 0; ctr < (NUM_CH * IMG_HEIGHT * IMG_WIDTH); ctr+=16) {

    //Fetch vector of 16 32-bit elements every iteration
    int16 vOut = *stov_ptr(int16, (int32_t*)&dataArray[ctr]);

    //Store vector in image buffer
    __SA0ADV(int16, &imageBuffer[0][0][0]) = vOut;

    //Display vector
    printf("vOut[%d] = ", ctr);
    vOut.print();
}
```

**Error -**

```
test9.cpp: In function ‘int main()’:
test9.cpp:59:46: error: no match for ‘operator=’ (operand types are ‘int16_ptr {
aka _c70_he_detail::vtype_ptr<int, 16>}’ and ‘int16 {aka _c70_he_detail::vtype<i
nt, 16>}’)
     __SA0ADV(int16, &imageBuffer[0][0][0]) = vOut;
                                              ^~~~
In file included from /home/kpit/psdk_rtos_auto_j7_06_01_00_15/ti-cgt-c7000_1.2.
0.STS/host_emulation/include/c7x_host_emulation.h:56:0,
                 from test9.cpp:3:
/home/kpit/psdk_rtos_auto_j7_06_01_00_15/ti-cgt-c7000_1.2.0.STS/host_emulation/i
nclude/vector.h:1264:7: note: candidate: _c70_he_detail::vtype_ptr<int, 16>& _c7
0_he_detail::vtype_ptr<int, 16>::operator=(const _c70_he_detail::vtype_ptr<int,
16>&) <deleted>
 class vtype_ptr : public t_ptr<vtype<ETYPE,N>,ETYPE,N>
       ^~~~~~~~~
```

## 4. Streaming address generator vector predicate store error

In “tutorial of vector predication and boundary awareness using streaming address generator(Chapter 3,Example 3 of SA)” error is thrown when SA engine tries to use the vector predicate and store in memory.

## Code -

```
//Open SE0
__SE0_OPEN((void *)&img_buffer[0], seTemplate);
//Open SA0
__SA0_OPEN(saTemplate);

//Fetch input data
for(int32_t ctr = 0; ctr < (IMG_HEIGHT * ceil(IMG_WIDTH/16.0)); ctr++) {

    //Fetch vector and advance (post-increment)
    int16 vIn = __SE0ADV(int16);

    //Read vector predicate from SA0
    vpred vpStore = __SA0_VPRED(int16);

    printf("\n vpStore[%d] = ", ctr);
    std::cout << std::hex;
    vpStore.print();

    //Use the vector predicate and store in memory
    __vstore_pred(vpStore, &__SA0ADV(int16, &cpy_buffer[0]), vIn);
}

//Close SE0
__SE0_CLOSE();
//Close SA0
__SA0_CLOSE();
```

## Error -

```
In file included from /home/kpit/psdk_rtos_auto_j7_06_01_00_15/ti-cgt-c7000_1.2.0.STS/host_emulation/include/c
7x_host_emulation.h:57:0,
                 from test10.cpp:3:
test10.cpp: In function ‘int main()’:
/home/kpit/psdk_rtos_auto_j7_06_01_00_15/ti-cgt-c7000_1.2.0.STS/host_emulation/include/c7x_he_strm.h:1027:78:
error: taking address of temporary [-fpermissive]
 #define __SA0ADV(type, baseptr) __sa_adv_##type(0, __sa_scale_##type, baseptr)
                                                                              ^
test10.cpp:105:29: note: in expansion of macro ‘__SA0ADV’
     __vstore_pred(vpStore, &__SA0ADV(int16, &cpy_buffer[0]), vIn);
                             ^~~~~~~~
test10.cpp:105:65: error: no matching function for call to ‘__vstore_pred(vpred&, int16_ptr*, int16&)’
     __vstore_pred(vpStore, &__SA0ADV(int16, &cpy_buffer[0]), vIn);
                                                             ^
```

## 5. Element Duplication error

In “tutorial for element duplication using streaming engine(Chapter 3,Example 2)” error is thrown while specifying element duplication flag.

**Code -**

```
//Structure of streaming engine flags
__SE_FLAGS seFlags;

//Initialize flags to default value so program only the flags you need
seFlags = __SE_FLAGS_default();

//Specify element type
seFlags.ELETYPE   = __SE_ELETYPE_32BIT;

//Specify lenght of vector to stream, in this case we want to stream a full lenght vector of 16-32bit elements
seFlags.VECLEN    = __SE_VECLEN_16ELEMS;

//Duplicate one 32-bit element 16 times to fill 512bit vector
seFlags.ELDUP = __SE_ELEDUP_16X
```

**Error -**

```
test12.cpp: In function ‘int main()’:
test12.cpp:38:9: error: ‘__SE_FLAGS {aka struct __SE_FLAGS_t}’ has no member nam
ed ‘ELDUP’; did you mean ‘ELEDUP’?
 seFlags.ELDUP = __SE_ELEDUP_16X;
         ^~~~~
         ELEDUP
```

**<u>This error is resolved by replacing seflags.ELDUP with seflags.ELEDUP</u>**

**Note –** Errors in Query 1,3,4 and 5 are also observed in while implementing “Tutorial of FIR filter using SE and SA (Chapter 2, Sample 4)”

## 6. Incorrect output while implementing circular streams.

In tutorial for “circular addressing using streaming engine (Chapter 3, Example 7 ) incorrect output is observed even after replicating the code in the tutorial.

## Observed output -

```
vIn[0] = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)
vIn[1] = (16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31)
vIn[2] = (32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47)
vIn[3] = (48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63)
vIn[4] = (64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79)
vIn[5] = (0, 0, -1426543712, 21866, 1780031584, 32765, -1429850925, 21866, 1, 0,
 1780035808, 32765, 1780031584, 32765, -1426543712, 21866)
vIn[6] = (0, 0, 0, 0, 1780038928, 32765, -1429883004, 21866, 0, 0, 256, 16, 16,
16, 16, 5)
vIn[7] = (1780031696, 32765, 0, 256, 131072, 262144, 131072, 0, 16, 16, 1, 16, 0
, 0, 0, 0)
vIn[8] = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)
vIn[9] = (16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31)
vIn[10] = (32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47)
vIn[11] = (48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63)
vIn[12] = (64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79)
vIn[13] = (0, 0, -1426543712, 21866, 1780031584, 32765, -1429850925, 21866, 1, 0
, 1780035808, 32765, 1780031584, 32765, -1426543712, 21866)
vIn[14] = (0, 0, 0, 0, 1780038928, 32765, -1429883004, 21866, 0, 0, 256, 16, 16,
 16, 16, 13)
vIn[15] = (1780031696, 32765, 0, 256, 131072, 262144, 131072, 0, 16, 16, 1, 16,
0, 0, 0, 0)
```

## Expected output -

```
vIn[0] = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)
vIn[1] = (16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31)
vIn[2] = (32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47)
vIn[3] = (48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63)
vIn[4] = (64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79)
vIn[5] = (80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95)
vIn[6] = (96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111)
vIn[7] = (112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127)
vIn[8] = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)
vIn[9] = (16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31)
vIn[10] = (32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47)
vIn[11] = (48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63)
vIn[12] = (64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79)
vIn[13] = (80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95)
vIn[14] = (96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111)
vIn[15] = (112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127)
```

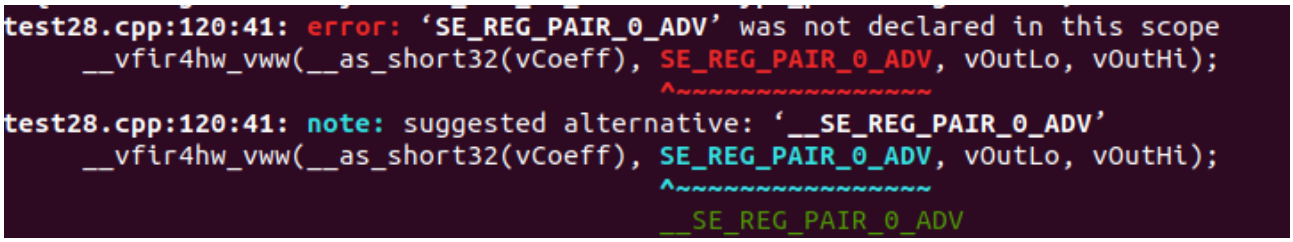
## 7. SE_REG_PAIR_0_ADV declaration error

In “tutorial of FIR filter using VFIR instruction (Chapter 2, Sample 5) ” an error is thrown in __vfir4hw_vww instruction, for declaration of SE_REG_PAIR_0_ADV.

**Code -**

```
//USE VFIR4HW to compute 1D FIR of 4 tap filter.
__vfir4hw_vww(__as_short32(vCoeff), SE_REG_PAIR_0_ADV, vOutLo, vOutHi);
```

**Error -**




**this error is resolved by replacing SE_REG_PAIR_0_ADV with __SE_REG_PAIR_0_ADV.**