```
          hsa_signal_query_acquire(queue->mailbox[i].full_flag,value);
      if(value.u64_value == 1){
        for(j=0; j < queue->mailbox[i].workgroup_size; j++) {
          printf("\n workitemid=%d computeunitid=%d\n",
                 queue->mailbox[i].workitem_id[j],
                 queue->mailbox[i].compute_unit_id[j]);
                 //here we can check PC, etc.
        }
        hsa_signal_send(queue_ptr->mailbox[i].full_flag, 0);
      }
    }
```

2.12 Agent Dispatch Support at the Core Level

2.13 Extensions to the Core Runtime API

HSA defines two broad types of extensions: HSA extensions and vendor extensions. Vendor extensions can further be provided as either single-vendor or multi-vendor extensions.

2.13.1 HSA Extensions

These are defined in the HSA runtime specification, but may not be supported by all vendors. HSA extensions have conformance tests defined by the foundation and are released with the same IP guidelines as the core runtime. The following enumeration defines the available HSA extensions:

```
typedef enum hsa_extension_t {
    HSA_EXTENSION_IMAGES,
    HSA_EXTENSION_DEBUG,
    HSA_EXTENSION_FINALIZER
}hsa_extension_t;
```

HSA_EXTENSION_IMAGES HSA extension to support images.

HSA_EXTENSION_DEBUG HSA extensions in support for providing information necessary to debug tools

HSA_EXTENSION_FINALIZER HSA extension for Finalizer that finalizes the brig to genrate compilation units that represent kernel and function code objects

Users can query to see if an extension is supported using:

```
hsa_status hsa_extension_query(hsa_extension_t extension);
```

extension input. The HSA extention that is being queried

This API returns HSA_STATUS_SUCCESS if the extension is supported. It returns HSA_STATUS_ERROR_EXTENSION_UNSUPPORTED if the extension is not available on the system.

All defines, structures, and API definitions for an HSA extension are defined in a specific header file.

HSA 1.0 includes: **hsa/extension_finalizer.h hsa/extension_images.h hsa/extension_debug.h**

When an implementor of the core runtime specification is not supporting any of the extension API, they will return HSA_STATUS_ERROR_EXTENSION_NOT_SUPPORTED as a return status for that API. This is so the user code can be successfully linked. Users are responsible for querying for the existence of the extension before calling API functions associated with that function.

2.13.2 Vendor Extensions

Vendors may define single-vendor extensions to HSA core runtime, or multiple vendors may collaborate to define an extension. The difference is in the naming scheme used for the symbols (defines, structures, functions, etc.) associated with the function:

- Symbols for single-vendor extensions that are defined in the global namespace must use the following naming convention:
  - *hsa_svext_<COMPANY_NAME>_*. For example, a company "Test" defining a single-vendor extension would use the prefix *hsa_svext_test_*. Company names must be registered with the HSA Foundation, must be unique, and may be abbreviated to improve the readability of the symbols.
- Symbols for multi-vendor extensions that are defined in the global namespace must use the following naming convention:
  - *hsa_mvext_* For example, if another company embraces extension in the example above from Company "Test", the resulting symbols would use the prefix *hsa_mvext_*.

Any constant definitions in the extension (#define/enumerations) use the same naming convention, except using all capital letters. So, using the single-vendor extension example from above, the associated defines and enumerations would have the prefix HSA_SVEXT_TEST_.

The symbols for all vendor extensions (both single-vendor and multi-vendor) are captured in the file **hsa/vendor_extensions.h**. This file is maintained by the HSA Foundation. This file includes the enumeration hsa_vendor_extension_t which defines a unique code for each vendor extension and multi-vendor extension. Vendors can reserve enumeration encodings through the HSA Foundation. Multi-vendor enumerations begin at the value of 1000000. For example, using the examples above, the hsa_vendor_extension_t enumeration might be:

```
typedef enum hsa_vendor_extension_s {
  HSA_SVEXT_START= 0,
  HSA_SVEXT_TEST_FOO = 1,
  HSA_SVEXT_TEST_ANOTHER_EXT = 2,
  HSA_MVEXT_START = 1000000,
  HSA_MVEXT_FOO   = 1000001
}hsa_vendor_extension_t;
```

HSA_SVEXT_START= 0 start of the single vendor extension range

HSA_SVEXT_TEST_FOO = 1 Company test, starts with FOO symbol

HSA_SVEXT_TEST_ANOTHER_EXT = 2 Company test has another_ext symbol

HSA_MVEXT_START = 1000000 multi vendor extension starts at 1000000

HSA_MVEXT_FOO = 1000001 multivendor extension has a symbol foo

HSA defines the following query function for vendor extensions:

```
hsa_status hsa_vendor_extension_query(hsa_vendor_extension_t
   extension, void *extension_structure);
```

extension input. The vendor extention that is being queried

This API returns `HSA_STATUS_SUCCESS` if the extension is supported. Additionally, *extension_structure* is written with extension-specific information such as version information, function pointers, and data values. **hsa/vendor_extension.h** defines a unique structure for each extension. If the vendor extension is not supported, `HSA_STATUS_ERROR_EXTENSION_UNSUPPORTED` is returned, and `extension_structure` is not modified.

2.13.3 Example Definition And Usage of an Extension

An example that shows a hypothetical single-vendor extension “Foo” registered by company “Test”. The example includes four defines and two API functions. Note the use the structure `hsa_svext_test_foo_t` and how this interacts with the **`hsa_query_vendor_extension`** API call.

```
//---
// Sample hsa/vendor_extensions.h
// Company name is "Test" and extension is "Foo"

#define HSA_SVEXT_TEST_MYDEFINE1 0x1000
#define HSA_SVEXT_TEST_MYDEFINE2 0x0100
#define HSA_SVEXT_TEST_MYDEFINE3 0x0010
#define HSA_SVEXT_TEST_MYDEFINE4 0x0001



// The structure which defines the version, functions, and data for
// the extension:
typedef struct hsa_svext_test_foo_s {
    int major_version;  // major version number of the extension.
    int minor_version;  // minor version number of the extension.


    // Function pointers:
    int (*function1) ( int p1, int *p2, float p3, int p4);
    int (*function2) ( int* p1, int p2);

    // Data:
    unsigned foo_data1;
} hsa_svext_test_foo_t;

main() {
    struct hsa_svext_test_foo_t testFoo;

    hsa_status_t status = hsa_query_vendor_extension(
       HSA_SVEXT_TEST_FOO,
        &testFoo);
    if (status == HSA_STATUS_SUCCESS) {
        (*(testFoo.function2))(0, 0);
    }
}
```