HCS12X – Accessing Data (advanced)

The present document describes how programmer can help the HCS12X compiler to generate the more optimal code for data access. It will cover following topics:

- Converting addresses (from logical to global, …)
- Defining a variable with logical addresses and access global
- Mixed addressing mode access to global variables

Note 1:
Information described in this technical note apply to SMALL (-Ms) and BANKED (-Mb) Memory model. They do not apply to LARGE (-Ml) memory model.

Note 2:
We usually recommend using SMALL memory model for application with less than 32Kb code and BANKED memory model otherwise. We do not recommend using LARGE memory model.

Note 3:
Please refer to technical note TN238 for basic usage of data definition with various addressing modes with and without banked memory.

Converting addresses (from Logical to Global, …)

There are runtime functions available to convert perform address conversion.
These functions are implemented in datapage.c and are used for instance when you are assigning a __rptr pointer to a __far pointer.

1. For instance following code snippet (variables defined in non-paged extended addressing area):

```
char data;
volatile char temp;

void func1(void) {
  char *__far  ptr;
  char *__rptr rptr;

  ptr = &data;
  rptr = ptr;

  temp = *rptr;
}
```

2. Generates following code:

```
   20:    ptr = &data;
0000 ce0000        [2]     LDX   #GLOBAL(data)
0003 c600          [1]     LDAB  #GLOBAL_PAGE(data)
   21:    rptr = ptr;
0005 160000        [4]     JSR   _CONV_GLOBAL_TO_LOGICAL
   22:
```

```
   23:    temp = *rptr;
0008 5b16          [2]     STAB  /*RPAGE*/22
000a a600          [3]     LDAA  0,X
000c 7a0000        [3]     STAA  temp
```

3. The run time function _CONV_GLOBAL_TO_LOGICAL is implemented in datapage.c. Further conversion routines are available there.

Normally a programmer should not care about these conversion routines. The compiler will use them when appropriate to convert addresses. If one really needs to use them we recommend to read carefully the comment in the file datapage.c for information on how to pass parameters and get returned value.

Defining a variable with logical addresses and access global

Even when an object is defined in an __RPAGE, __EPAGE or __SHORT segment, the object can be accessed using its global address using a simple __far pointer.
The example below shows how to define a variable in a _RPAGE segment and use a __far pointer to access it.

1. Definition of a variable in a RPAGE segment and usage of a global pointer is straightforward:

```
#pragma DATA_SEG __RPAGE_SEG PAGED_RAM
char data[10] = {
   0x10, 0x20, 0x30, 0x40, 0x50
};
#pragma DATA_SEG DEFAULT

volatile char temp;

void func1(void) {
  char *__far  ptr;

  ptr = data;

  temp = *ptr;
}
```

2. Generates following code:

```
   29:    ptr = data;
0000 ce0000        [2]     LDX   #GLOBAL(data)
0003 c600          [1]     LDAB  #GLOBAL_PAGE(data)
   30:
   31:    temp = *ptr;
0005 5b10          [2]     STAB  /*GPAGE*/16
0007 18a600        [4]     GLDAA 0,X
000a 7a0000        [3]     STAA  temp
```

3. Defining the variable with global addressing and access with a logical pointer works accordingly.

Mixed addressing mode access to global variables

Around accessing a specific variable once with logical address, once with global address, you can take advantage of ANSI C cast operators. This solution does not need pointer variables.

1. For instance following code snippet:

```
#pragma DATA_SEG __RPAGE_SEG PAGED_RAM
char data[10] = {
   0x10, 0x20, 0x30, 0x40, 0x50
};
#pragma DATA_SEG DEFAULT

volatile char temp;

void func1(void) {

  temp = *((char *__far)data);
  temp = *((char *__rptr)data);
}
```

2. Generates following code:

```
   20:    temp = *((char *__far)data);
0002 ce0000        [2]     LDX   #GLOBAL(data)
0005 c600          [1]     LDAB  #GLOBAL_PAGE(data)
0007 5b10          [2]     STAB  /*GPAGE*/16
0009 18a600        [4]     GLDAA 0,X
000c 7a0000        [3]     STAA  temp
   21:    temp = *((char *__rptr)data);
000f c600          [1]     LDAB  #PAGE(data)
0011 5b16          [2]     STAB  /*RPAGE*/22
0013 f60000        [3]     LDAB  data
0016 7b0000        [3]     STAB  temp
```

3. So the same data is accessed once using global address and once using its logical address without the need of a pointer variable (please note: line 20 of this sample produces the same code as the example above, lines 29 and 31).

4. Defining the variable with global addressing mode

```
#pragma DATA_SEG __GPAGE_SEG PAGED_RAM
char data[10] = {
   0x10, 0x20, 0x30, 0x40, 0x50
};
#pragma DATA_SEG DEFAULT
```

5. The same code generates the following:

```
   23:    temp = *((char *__far)data);
0000 c600          [1]     LDAB  #GLOBAL_PAGE(data)
0002 5b10          [2]     STAB  /*GPAGE*/16
0004 18f60000      [4]     GLDAB data
0008 7b0000        [3]     STAB  temp
```

```
 24:    temp = *((char *__rptr)data);
000b ce0000        [2]     LDX   #data
000e c600          [1]     LDAB  #PAGE(data)
0010 5b16          [2]     STAB  /*RPAGE*/22
0012 a600          [3]     LDAA  0,X
0014 7a0000        [3]     STAA  temp
```