RAA271005 I2C Register Paging

This App Note describes the implementation of register paging in RAA271005 Regulation and Protection Blocks. Reading of page registers is not required nor recommended. Using CRC is recommended for the correctness of a write to any register. If the customers expect to read updated value of page dynamically, this app note can be referenced for the correct software implementation.

Contents

1. Regulation Register Paging 1

2. Recommended steps to change page before every R/W access to the target register 2

3. Examples of paging between the Regulation Registers 3

4. Example Application Software Algorithm 4

5. Revision History 5

1. Regulation Register Paging

RAA271005 regulation and protection register map contains 512 registers in each block. For simplifying the access to these registers using single byte addressing, RAA271005 implements paging. In regulation block, paging is handled by Page selection registers IO_PAGE_REGU(0x000), IO_PAGE1(0x100) and IO_PAGE2(0x200). These registers change the page immediately when written but are not designed to synchronize automatically on the write operation and cannot be read unless they are written with current page information. These registers can be used for keeping track of the current page by the application software by synchronizing these registers on every page change.

Note: This app note only talks about regulation register paging, protection paging implementation is different and does not require same process as described. IO_PAGE1(0x100) and IO_PAGE2(0x200) do not exist in protection block and page selector is implemented such that the page information is updated automatically on every write.

2. Recommended steps to change page before every R/W access to the target register

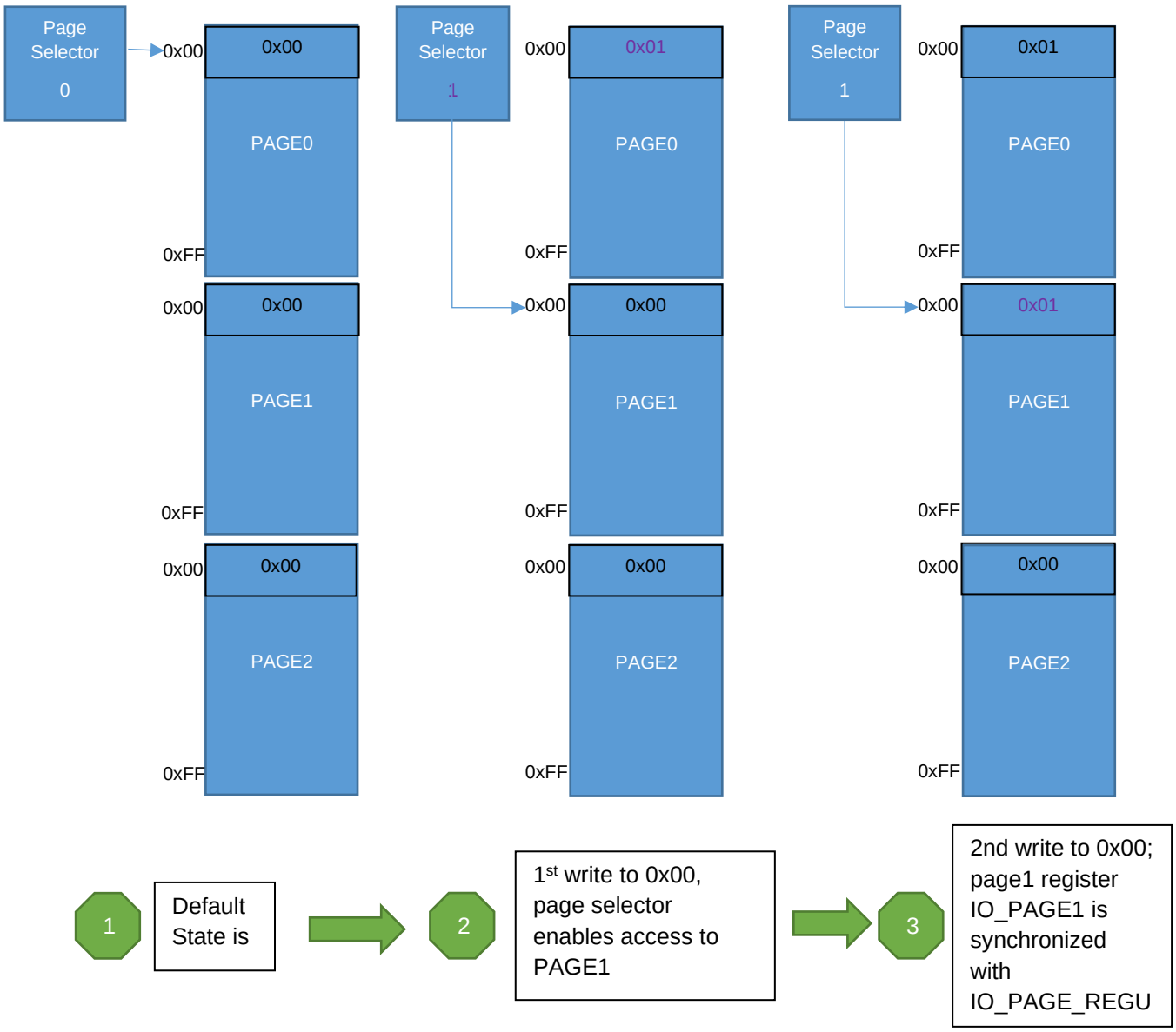


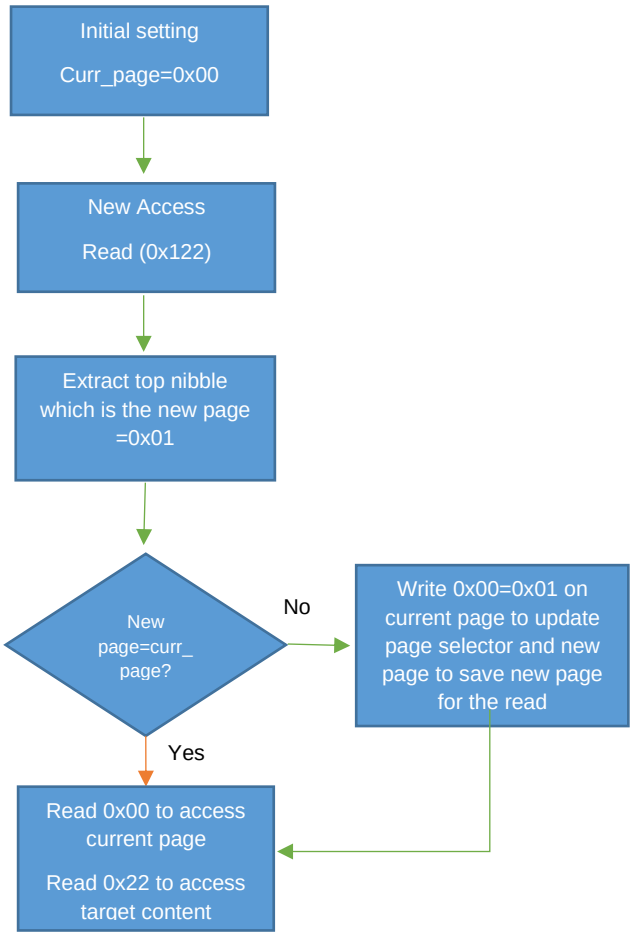
Fig1: Steps to change the Regulation PAGE

3. Examples of paging between the Regulation Registers

Current Page	New Access to Page	Next Transactions
Page0	Page0	No page change needed. If the new access is to the current page, no page change or update commands are needed, and the target register can be accessed directly. Same applies for Page1/Page2.
Page0	Page1/Page2	1. Write IO_PAGE_REGU 0x00 with 0x01/0x02 to change page selector to access Page1/Page2. Page Selector is updated immediately 2. Write IO_PAGE1/IO_PAGE2 0x00 with 0x01/0x02 to keep current page information updated for the read. 3. Write/Read the content to Page1/Page2 register that needs to be accessed
Page1/Page2	Page0	1. Write IO_PAGE1/IO_PAGE2 0x00 with 0x00, to change page selector to access Page0. Page Selector is updated immediately 2. Write IO_PAGE_REGU 0x00 with 0x00 to keep current page information updated for the read. 3. Write/Read the content to Page0 register that needs to be accessed
Page1	Page2	1. Write IO_PAGE1 0x00 with 0x02, to change page selector to access Page2. Page Selector is updated immediately 2. Write IO_PAGE2 0x00 with 0x02 to keep current page information updated for the read. 3. Write/Read the content to Page2 register that needs to be accessed
Page2	Page1	1. Write IO_PAGE2 0x00 with 0x01, to change page selector to access Page1. Page Selector is updated immediately 2. Write IO_PAGE1 0x00 with 0x01 to keep current page information updated for the read. 3. Write/Read the content to Page1 register that needs to be accessed

4. Example Application Software Algorithm

Following example talks about reading a page1 register when the current page selector is at Page0.



Pseudo Code:

```
Curr_Page=0x00                  //Initial state stored in application software
Read(0x122)                     //New access
New_Page=((0x122>>8)&0xFF)      //extract new page information from the new access
If(New_Page!=Curr_Page)         //Determine if page change is needed
{
   Write(0x00,0x01)             //change the page selector
   Write(0x00,0x01)             //Page is changed, write the page selector
   Read(0x00)                   //To check the page is changed
   Read(0x22)                   //Access content from target register
}
Else
{
   Read(0x22)                   //Access content from target register
}
```

5. Revision History

Revision	Date	Description
1.0	May24, 2022	Initial release.