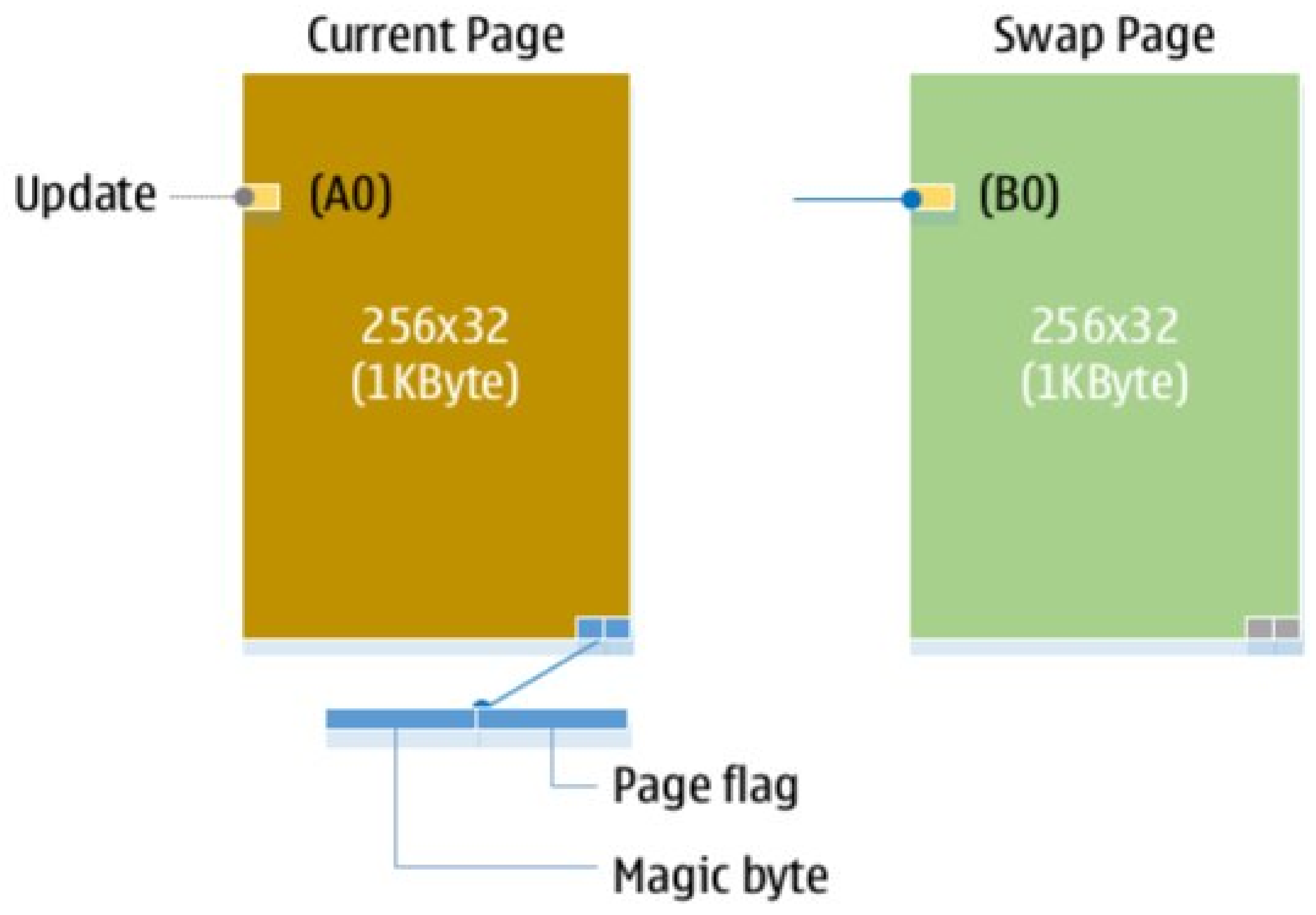
In terms of, The E2PCTL controller improves efficiency by implementing a continuous data update mode that reduces redundant erasing during data updates. The controller manages each page separately. The last 2 bytes of each page is used to store information about the page status. Therefore, if the E2PROM emulation partition is larger than 1K, the user must pay special attention to these two special addresses when crossing the 1K Page space. The last 2 bytes of each 1K space are reserved for E2PCTL use, the user cannot read and write these 2 bytes.

E2PCTL page-based exchange algorithm:



\

As illustrated, the E2PCTL uses two 1K byte Pages of FLASH to emulate a single page of E2PROM Memory. One of these two pages is marked as the current page, and the other is the exchange page. The E2PCTL uses the last 2 bytes of each Page to store Page information. When we need to update a byte in the page, such as the 0xA0 byte in the illustration above. First, the Swap Page is erased, but not the Current Page. The next step is the definition of the Current Page data addresses from before and after the updated data as CP0 and CP1 respectively. This is where the E2PCTL takes over. It will copy the CP0 data to the corresponding addresses of the exchange page. Next it writes the updated data that triggered this entire process. This is written to the Swap Page at the address that corresponds to the address to be changed (B0), and lastly CP1 is copied to the exchange page.

After completing the above operation, the data has been exchanged to the Swap Page, but the page status has not been changed. If a power failure occurs at any time up to this point in the operation, the Current Page data is unaffected, which ensures the integrity of the data. If all goes well, the last step the E2PCTL controller does is change the page status byte at the very end of CP1 on the Swap Page. This action sets the Swap Page as the Current Page. The E2PCTL page exchange process is shown in the figure on the following page (1→2→3→4):