Frequently Asked Questions

Title: Differences between Px and Px-1760
Date: 07 Feb 2021
Card/Board/Module: 1553Px cards & modules
Operating System: All

Question:

What is the difference between the regular version of M4K1553Px[S] / M8K1553Px[S] module, and the version with the -1760 suffix ?

Answer:

The 1760 version of the module is a superset of the 1553 version. That is, all of the features of 1553 exist in the 1760 module, plus there are a few additional features that exist only in the 1760 version. The API is the same, with some additional functions that implement the **additional 1760 features**. These features are:

1. **Header Word**
Some SubAddresses (1,11,14) require that the first data word be a specific value (in specific cases). These are listed in the manual (hardware & software), under sections called "Header Word" or "1760 Options".
• This is designed as per the 1760 specification.
• We allow you to disable/suppress this requirement, using function **Set_Header_Exists_Px**, with parameter HEADER_DISABLE.

2. **Checksum**
In **BC mode**, you can set a message to require a checksum as the last data word, as defined in the manuals (using function **Enable_Checksum_Px**). The module will calculate the checksum and place it as the last data word in the BC2RT message, or check if the received checksum in an RT2BC message is correct.
[You can also set that the checksum for a BC2RT message should have an error injected into it (using function **Enable_Checksum_Error_Px**).]

In **RT mode**, you can set that certain datablocks (for Rx or Tx) can be designated to require a checksum in the last data word (using function **Set_Checksum_Blocks_Px**). A datablock can be associated with any RTid (Rx or Tx), using function **Assign_RT_Data_Px**.

In **Monitor mode**, the module always looks for the checksum and sets a flag in the message status word if the checksum word is not correct.

Here is sample code to call these functions:

----
for BC mode:

```
   // Enable_Checksum_Px();
   msgentry = 0;
   enable = ENABLE;  // or DISABLE;
   status = Enable_Checksum_Px(handle, frameid, msgentry, enable);
   if (status < 0)
   {
      printf("Enable_Checksum_Px returned error: %s\n",Print_Error_Px(status));
      printf("Press any key to exit the program ... ");
      _getch();
   }
   else
      printf("Enable_Checksum_Px returns status value %d\n", status);

   // Enable_Checksum_Error_Px();
   msgentry = 1;
   enable = ENABLE;  // or DISABLE;
   status = Enable_Checksum_Error_Px(handle, frameid, msgentry, enable);
   if (status < 0)
   {
      printf("Enable_Checksum_Error_Px returned error: %s\n",Print_Error_Px(status));
      printf("Press any key to exit the program ... ");
      _getch();
   }
   else
      printf("Enable_Checksum_Error_Px returns status value %d\n", status);
```

----
for RT mode:

```
   // Set_Checksum_Blocks_Px();
   int csum_blocks=50;
   status = Set_Checksum_Blocks_Px(handle, csum_blocks);
   if (status < 0)
   {
      printf("Set_Checksum_Blocks_Px returned error: %s\n",Print_Error_Px(status));
      printf("Press any key to exit the program ... ");
      _getch();
   }
   else
      printf("Set_Checksum_Blocks_Px returns status value %d\n", status);
```