The MIDI protocol consists of a set of commands - note ons, note offs, controllers, program changes etc. - followed by data for that command. This information is passed in bytes - an 8-bit package. A bit can be either ON or OFF [1 or 0] giving a total of 256 [2^8] permutations for a byte.

An on bit has an associated current flow, while an off bit has not. [Think of a telegraph signal]

The protocol states that if the first bit in a byte is 1, then the byte is a command, whereas if it is 0, then the byte contains data for the previous command. If the first bit is 1, then the next 3 bits define the type of command - note on, note off, controller, program change etc.. [8 types... 2*2*2] The last 4 bits in a command byte define the midi channel - 2*2*2*2 - therefore 16 midi channels.

For a note-on command, an example of the command byte would be 10010111 - first bit command, next three for note-on, last 4 means its going on channel 7 :: 0111 bin = 7 dec
Note on command bytes are followed by 2 data bytes :: the leading bit of each must be 0, as stated above. The first data byte tells which note to be played, and the second data byte tells what velocity it is to be played at .
The possible range of values for these data bytes is from a value of 0 to a value of 127 –
ie from [0]000 0000 to [0]111 1111.

So, for a note on command, we get three bytes in sequence saying "Command here.....play a note....play it on channel [1 to 16], this is the note number [0 to 127] and this is the velocity [0 to 127]".

And that, in essence, is how all MIDI messages will present themselves - a command byte, and a packet of data relevant to that command.

<u>Program change command:</u>

In the case of PROGRAM CHANGES, the format is Command byte - 1 100 xxxx - bit 1 says command, bits 2~4 say command is PROGRAM CHANGE, bits 5~8 define the midi channel.
This command byte is followed by 1 data byte - format 0 xxxxxxx - 0 for data byte, bits 2~8 [2*2*2*2*2*2*2] give a range of 128.
So, when the MIDI protocol was formalised, it allowed for a program change command with a data range of 0 to 127.

Remember that this was in the early 80's, when synths had only a few - if any - slots for storing a memory.
With time, memory in synthesisers and tone generators increased exponentially, and soon surpassed the 128 patches which could be called up via MIDI.

<u>Controller Command to the rescue:</u>

Because there were a growing number of controllers mooted when the MIDI protocol was designed, the controller format was decided as :

Byte 1 - Command [Controller] byte :: Byte 2 - Data byte - Controller number :: Byte 3 - Data Byte value for that controller [much like note-on/which note/velocity value]

eg <u>10110001</u> <u>00000111</u> <u>01111111</u> - Command Byte/Controller command/Midi Channel 01 :: Controller No 7 :: value 127

The protocol thus allowed for 128 controllers – numbered from Controller 0 to Controller 127
Controller No 1 is normally for Modulation – the Mod wheel on your synth sends out Controller 1 info
Controller No 7 is for Volume Control – the volume pedal on your synth sends out Controller 7 info.

Now while there were maybe 20 different controllers types available, this meant that there were lots of slots left in the range of 0 to 127. These controller slots were called **UNDEFINED** at the time of the MIDI format inception, but with a view to being allocated later as requirements dictated.

As manufacturers became aware of the insufficient slots in the program change command structure, so they came realised that something had to be done to increase the range of program changes available.

Parallel to this, there was another set of program change methodologies immerging. Roland was using a patch system which had 8 slots – something like A 1 thru A8, then B1 thru B8, C1 thru C8 etc. A similar 8-based setup could be 11 thru 18 then 21 thru 28 then 31 thru 38 etc. And some manufacturers were using the good old decimal system….patch 1 thru whatever, each patch incrementing by 1.

These methodologies were known as bank systems – in the case of Roland, each bank had 8 patches.
As time elapsed, Roland progressed to having 128 patches in each bank- obviously because you could address 128 individual patches by a program change command. So now you had a bank with 128 patches – the need now was to be able to command a BANK CHANGE to address a patch outside that initial 128.

At this time, manufacturers realised they had probably erred on the side of minimalism whe designing the protocol…..there was such a huge increase in the potential for electronic music instruments that their initial projections were way too conservative. So when it came to providing a method for Bank Changes, they decided:

To use TWO undefined controllers – these were allocated to defining patch Banks.
These are Controller 0 and Controller 32 – Controller 0 is known as BankSel MSB and 32 as BankSel LSB
[MSB stands for Most Significant Byte :: LSB for Least Significant Byte]
The combination of these two controllers means you can address 128 * 128 BANKS !! Each bank can have 128 patches!!
That is, you can address 128 * 128 * 128 patches via this method – **2097152** patches!!

This is still way beyond the scope of any instrument [at least ones that I am aware of!], but it means we won't be caught out for a patch slot any time soon!!!