<u>Hamming Code</u>

The use of simple parity allows detection of single bit errors in a received message. Correction of such errors requires more information, since the position of the bad bit must be identified if it is to be corrected. (If a bad bit can be found, then it can be corrected by simply complementing its value.) Correction is not possible with one parity bit since any bit error in any position produces exactly the same information - "bad parity".

If more bits are included with a message, and if those bits can be arranged such that different error bits produce different error results, then bad bits could be identified. In a 7-bit message, there are seven possible single bit errors, so three error control bits could potentially specify not only that an error occured but also which bit caused the error.

Similarly, if a family of codewords is chosen such that the minimum distance between valid codewords is at least 3, then single bit error correction is possible. This distance approach is "geometric", while the above error-bit argument is 'algebraic'.

Either of the above arguments serves to introduce the Hamming Code, an error control method allowing correction of a single bit error.

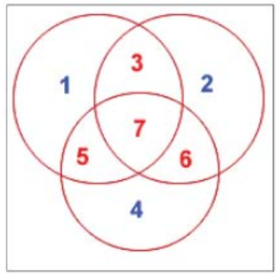
The Hamming Code

Consider a message having four data bits (D) which is to be transmitted as a 7-bit codeword by adding three error control bits. This would be called a (7,4) code. The three bits to be added are three EVEN Parity bits (P), where the parity of each is computed on different subsets of the message bits as shown below.

7	6	5	4	3	2	1	
D	D	D	P	D	P	P	7-BIT CODEWORD
D	-	D	-	D	-	P	(EVEN PARITY)
D	D	-	-	D	P	-	(EVEN PARITY)
D	D	D	P	-	-	-	(EVEN PARITY)

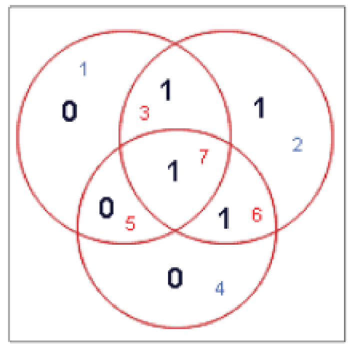
Why Those Bits? - The three parity bits (1,2,4) are related to the data bits (3,5,6,7) as shown at right. In this diagram, each overlapping circle corresponds to one parity bit and defines the four bits contributing to that parity computation.

For example, data bit 3 contributes to parity bits 1 and 2. Each circle (parity bit) encompasses a total of four bits, and each circle must have EVEN parity. Given four data bits, the three parity bits can easily be chosen to ensure this condition. It can be observed that changing any one bit numbered 1..7 uniquely affects the three parity bits. Changing bit 7 affects all three parity bits, while an error in bit 6 affects only parity bits 2 and 4, and an error in a parity bit affects only that bit. The location of any single bit error is determined directly upon checking the three parity circles.



For example, the message 1101 would be sent as 1100110, since:

7	6	5	4	3	2	1	
1	1	0	0	1	1	0	7-BIT CODEWORD
1	-	0	-	1	-	0	(EVEN PARITY)
1	1	-	-	1	1	-	(EVEN PARITY)
1	1	0	0	-	-	-	(EVEN PARITY)



When these seven bits are entered into the parity circles, it can be confirmed that the choice of these three parity bits ensures that the parity within each circle is EVEN, as shown here.

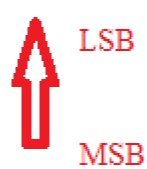
It may now be observed that if an error occurs in any of the seven bits, that error will affect different combinations of the three parity bits depending on the bit position.

For example, suppose the above message 1100110 is sent and a single bit error occurs such that the codeword 1110110 is received:

```
    transmitted message                 received message
     1 1 0 0 1 1 0        ----------->     1 1 1 0 1 1 0
BIT: 7 6 5 4 3 2 1                    BIT: 7 6 5 4 3 2 1
```

The above error (in bit 5) can be corrected by examining which of the three parity bits was affected by the bad bit:

7	6	5	4	3	2	1			
1	1	1	0	1	1	0	7-BIT CODEWORD		
1	-	1	-	1	-	0	(EVEN PARITY)	NOT!	1
1	1	-	-	1	1	-	(EVEN PARITY)	OK!	0
1	1	1	0	-	-	-	(EVEN PARITY)	NOT!	1



In fact, the bad parity bits labeled 101 point directly to the bad bit since 101 binary equals 5. Examination of the 'parity circles' confirms that any single bit error could be corrected in this way.

The value of the Hamming code can be summarized:

1. Correction of single bit errors;
2. Cost of 3 bits added to a 4-bit message.
3. The ability to correct single bit errors comes at a cost which is less than sending the entire message twice.
4. Detection of 2 bit errors (assuming no correction is attempted).