JOINING FORCES

We introduced the principles of networking on micros on page 321. Now we look at a game application for the cheapest network of all — the Sinclair ZX Net. This system is the simplest possible but its potential for both serious and amusing use should not be overlooked.

Battleships is a classic pen and paper game. Each player has two grids, which he keeps hidden from his opponent. On one he marks his own ships, on the other he marks his progress as he 'fires at' — that is, tries to guess the position of — his opponent's ships.

The program we have developed works on two Sinclair Spectrums linked together with a network. Both Spectrums must be equipped with Interface 1. Each player sits at his own screen and the two computers send each other messages that report where the players are shooting and what the results are.

The first hazard you meet is that of identifying the players. In order to communicate, each Spectrum has a different network station number. The two Spectrums start off with identical programs but somehow must end up with different station numbers. This is handled automatically by a routine at line 2000. When the program is RUN, both machines will claim to be station 1 on the public 'broadcast' channel.

Whichever machine is RUN first will become station 1 and the other machine will then make itself station 2. This system works well for Battleships. The program then assumes that whoever is station 1 is player 1 and therefore allows him to shoot first. However, if the two programs are started within a fraction of a second of each other, the two messages "I'm station number 1" simply collide and the ZX Net system will stop functioning until the players press the BREAK key.

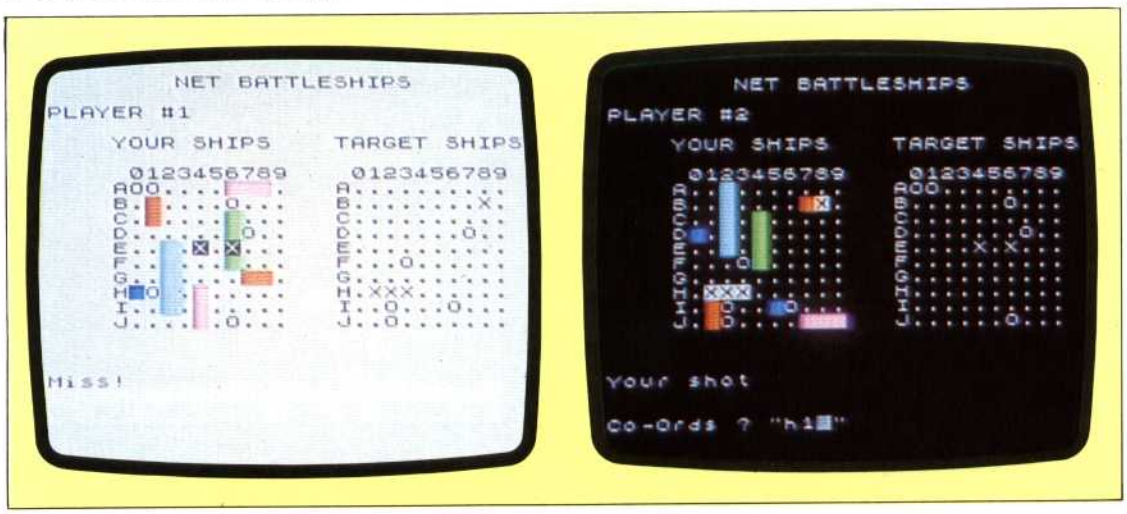
Once you know who's who, it is easy for the program to communicate with its opposite number across the network. While one player is picking a target square on his machine, the other is waiting to receive his choice. The machines will then swap over. One machine calculates the results of the shot and sends back a message while the other waits to receive the results and update its screen display accordingly.

Provided you make sure that the two programs always 'fit' together -- one sending, the other receiving — this is very easy to program. You don't have to worry about timing, sending messages too late, or missing them after they've been sent because ZX Net stops until both stations are ready and then transmits the data. So it doesn't matter if one player takes a long time to select a target or if the program takes a long time to update its screen.

Another point worth noting is that the amount of data being transmitted should be kept to a minimum. There's no need to send long chunks of data. Provided both programs know what the information means, you can communicate using short codes. In Battleships, the program returns the result of a shot as a two character string. The first character is a code:

1 Miss
2 Hit a ship
3 Hit and sunk a ship
4 Hit and sunk a ship and won the game

The second character is the class of ship that was hit (or a 0 for a miss). The program at the other end can decode this information and display appropriate messages. This method makes the time taken to execute each turn so fast you wouldn't think that another computer was involved at all.



LIZ HEANEY

Fleet Action
Each player's fleet comprises ships of different sizes (from motor torpedo boat to carrier) placed anywhere in the grid. The screen displays the position and status of one player's ships and the enemy's shots at them, and the position and effect of his shots at the enemy's ships: "X" shows a hit, "O" for a miss