--- The concept:

Make spectator coordinates, angle, pitch and player number available in TNS ACS scripts by inserting these values into the puke command arguments on the client side, before the command gets transmitted to the server through rcon.

Approach 1: Use read-only CVARs or simple labels as puke arguments to be substituted with chosen values.

Approach 2: Use a format string to make the solution more general and more variable.

For both: Utilize the whole length of a puke argument (32 bits) to transmit useful data.

In addition: Make the puke command accept CVARs as arguments in general.

--- Values and their representation:

each coordinate (16 bits, signed short)
angle (8 bits, unsigned char)
player number (8 bits, unsigned char)
pitch (16 bits, signed short)

--- The variables of the most basic solution for TNS (illustrative names):

$yx (y coordinate, x coordinate)
$Naz (own player number, angle, z coordinate)
$Nnp (own player number, player number, pitch)

--- Value origin:

All values are for the current viewpoint, ie when spectating someone, values of the spectated player (including player number) are used. The only exception being the "own player number" which is always the number of the client the command was executed from.

--- Approach 2 overview

A format string that will allow custom placement and combination of predefined fetched variables/labels within a puke argument and even mixing these with constants up to the limit of 32 bits.

--- Format string parser behaviour

A special charachter (%) denotes the start of each predefined label and constant type specifier. If the puke argument starts with a special character, the argument is considered to be a format string and it is run through the parser, if not, the parser is bypassed completely.

The character directly folowing the special character denotes a predefined label or a constant type specifier. Any other character than those mentioned in the predefined labels and the constant type specifiers sections below will generate a parse error.

Up to 11 characters (10 digits to represent an int and an optional sign) following a constant type specifier is read and then converted to int. If the next special character (%) or the end of the format string is not reached within these 11+1 characters, a parse error will be generated. The bits that would exceed the size of the specified type are cut off before they are added to the argument.

With predefined labels, the trailing characters are ignored but a parse error will still be generated if more than 11 of those follow.

When a parse error occurs, the argument is set to 0.

--- Predefined labels (see value representation above)

%x - x coordinate
%y - y coordinate
%z - z coordinate
%a - angle
%p - pitch
%n - player number
%N - own player number

--- Constant type specifiers (see format string examples for usage information):

%c - char
%C - unsigned char
%s - short
%S - unsigned short
%i - int

--- Examples of format strings:

Single label label-only format string:

%n

Multi-label label-only format string:

%y%x
%N%n%z

Single constant constant-only format string:

%i846545 (equivalent to an ordinary argument)
%s-15684 (same as above but limited to 16 bit signed values)
%C255 (same as above but limited to 8 bit unsigned values)

Multi-constant constant-only format string

%s-32456%S65231
%c-128%C165%s1522

Mixed format string:

%z%s0
%S32998%n%c-12
%s-1315%p

--- General CVAR parsing

The parser can be invoked even for $ as a special character and branched at the beginning to substitute the CVAR name with its value. Everything that follows after the $ character would be the CVARs name. It might be preferable for you to do this on a higher level – just keep in mind that the CVAR value needs to be inserted on the client, before rcon sends the command to the server.