crc.generator

Construct CRC generator object

Syntax

h = crc.generator(polynomial)

h = crc.generator(detectorObj)

h = crc.generator(‘Polynomial’, polynomial, ‘param1’, val1, etc.)

h = crc.generator

Description

h = crc.generator(polynomial) constructs a CRC generator object H defined by the generator polynomial POLYNOMIAL.

h = crc.generator(detectorObj) constructs a CRC generator object H defined by the parameters found in the CRC detector object DETECTOROBJ.

h = crc.generator(‘property1’, val1, ...) constructs a CRC generator object H with properties as specified by the PROPERTY/VALUE pairs.

h = crc.generator constructs a CRC generator object H with default properties. It constructs a CRC-CCITT generator, and is equivalent to: h = crc.generator('Polynomial', '0x1021', 'InitialState', '0xFFFF', ...

'ReflectInput', false, 'ReflectRemainder', false, 'FinalXOR', '0x0000').

Properties

The following table describes the properties of a CRC generator object. All properties are writable, except Polynomial.

Property	Description
Polynomial	The generator polynomial that defines connections for a linear feedback shift register. This property can be specified as a binary vector representing descending

Property	Description
	powers of the polynomial. In this case, the leading '1' of the polynomial must be included. It can also be specified as a string, prefaced by '0x', that is a hexadecimal representation of the descending powers of the polynomial. In this case, the leading '1' of the polynomial is omitted.
InitialState	The initial contents of the shift register. This property can be specified as a binary scalar, a binary vector, or as a string, prefaced by '0x', that is a hexadecimal representation of the binary vector. As a binary vector, its length must be one less than the length of the binary vector representation of the Polynomial.
ReflectInput	A Boolean quantity that specifies whether the input data should be flipped on a bytewise basis prior to entering the shift register.
ReflectRemainder	A Boolean quantity that specifies whether the binary output CRC checksum should be flipped around its center after the input data is completely through the shift register.
FinalXOR	The value with which the CRC checksum is to be XORed just prior to being appended to the input data. This property can be specified as a binary scalar, a binary vector, or as a string, prefaced by '0x', that is a hexadecimal representation of the binary vector. As a binary vector, its length must be one less than the length of the binary vector representation of the Polynomial.

CRC Generation Algorithm

For information pertaining to the CRC generation algorithm, refer to the CRC Non-Direct Algorithm section of the Communications System Toolbox™ User's Guide.

Generator Method

encoded = generate(h, msg) generates a CRC checksum for an input message using the CRC generator object H. It appends the checksum to the end of

MSG. The binary-valued MSG can be either a column vector or a matrix. If it is a matrix, then each column is considered to be a separate channel.

Usage Example

The following examples demonstrate the use of this object.

```
% Construct a CRC generator with a polynomial defined
% by x^4+x^3+x^2+x+1:
h = crc.generator([1 1 1 1 1])
% Construct a CRC generator with a polynomial defined
% by x^4+x^3+x^2+x+1, all-ones initial states, reflected
% input, and all-zeros final XOR value:
h = crc.generator('Polynomial', '0xF', 'InitialState', ...
'0xF', 'ReflectInput', true, 'FinalXOR', '0x0')
% Create a CRC-16 CRC generator, then use it to generate
% a checksum for the
% binary vector represented by the ASCII sequence '123456789'.
gen = crc.generator('Polynomial', '0x8005', ...
'ReflectInput', true, 'ReflectRemainder', true);
% The message below is an ASCII representation of ...
% the digits 1-9
msg = reshape(de2bi(49:57, 8, 'left-msb')', 72, 1);
encoded = generate(gen, msg);
% Construct a CRC generator with a polynomial defined
% by x^3+x+1, with zero initial states,
% and with an all-ones final XOR value:
h = crc.generator('Polynomial', [1 0 1 1], ...
                  'InitialState', [0 0 0], ...
                  'FinalXOR', [1 1 1])
```