# CalcPauliStringRetargeted

```
SetDirectory @ NotebookDirectory[];
Import["../Link/QuESTlink.m"];
```

## Doc

```
? GetPauliStringRetargeted
```

Symbol

GetPauliStringRetargeted[string, rules] returns the given Pauli string but with its target qubits modified as per the given rules. The rules can be anything accepted by ReplaceAll.

For instance GetPauliStringRetargeted[..., {0–>1, 1–>0}] swaps the first and second qubits, and GetPauliStringRetargeted[..., q_ –> q + 10] shifts every qubit up by 10.

This function modifies only the qubits in the Pauli string and avoids modifying coefficients, so it is a safe alternative to simply evaluating (string /. rules).

## Correctness

```
GetPauliStringRetargeted[X_2, 2 → 3]
GetPauliStringRetargeted[2 X_2, 2 → 3]
```

$X_3$

$2\ X_3$

```
GetPauliStringRetargeted[3 X_3 Z_5, {3 → 5, 5 → 4}]
```

$3\ X_5\ Z_4$

```
GetPauliStringRetargeted[X_3 + X_4, {3 → 4}]
```

$2\ X_4$

```
GetPauliStringRetargeted[3 X_3 Z_5 + Y_0 + 5 X_2 Z_4 + Z_5 X_2, {3 → 5, 5 → 4}]
```

$Y_0 + 6\ X_2\ Z_4 + 3\ X_5\ Z_4$

```
GetPauliStringRetargeted[a X_0 Y_1 Z_2 + b X_3 Y_4 Z_5, q_ :→ q + 10]
```

$a\ X_{10}\ Y_{11}\ Z_{12} + b\ X_{13}\ Y_{14}\ Z_{15}$

# Errors

**GetPauliStringRetargeted[2, 0 → 1]**

GetPauliStringRetargeted: Invalid arguments. See ?GetPauliStringRetargeted

$Failed

**GetPauliStringRetargeted[$X_0 + 2$, 0 → 1]**

GetPauliStringRetargeted: Invalid arguments. See ?GetPauliStringRetargeted

$Failed

**GetPauliStringRetargeted[$X_2$, notamap]**

GetPauliStringRetargeted: Invalid rules caused the below ReplaceAll error:

ReplaceAll: {notamap} is neither a list of replacement rules nor a valid dispatch table, and so cannot be used for replacing.

$Failed

**GetPauliStringRetargeted[$X_0\, Y_0$, 0 → 1]**

GetPauliStringRetargeted: Invalid arguments. See ?GetPauliStringRetargeted

$Failed

**GetPauliStringRetargeted[$X_{-1}$, 0 → 1]**

GetPauliStringRetargeted: Invalid arguments. See ?GetPauliStringRetargeted

$Failed