

**potgo**

<b>COLLABORATORS</b>
----------------------

	<i>TITLE :</i> potgo		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 19, 2024	

<b>REVISION HISTORY</b>
-------------------------

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>potgo</b>	<b>1</b>
1.1	potgo.doc . . . . .	1
1.2	potgo.resource/AllocPotBits . . . . .	1
1.3	potgo.resource/FreePotBits . . . . .	2
1.4	potgo.resource/WritePotgo . . . . .	2

# Chapter 1

## potgo

### 1.1 potgo.doc

```
AllocPotBits()  
FreePotBits()  
WritePotgo()
```

### 1.2 potgo.resource/AllocPotBits

#### NAME

AllocPotBits -- Allocate bits in the potgo register.

#### SYNOPSIS

```
allocated = AllocPotBits(bits)  
D0                                D0
```

```
UWORD AllocPotBits( UWORD );
```

#### FUNCTION

The AllocPotBits routine allocates bits in the hardware potgo register that the application wishes to manipulate via WritePotgo. The request may be for more than one bit. A user trying to allocate bits may find that they are unavailable because they are already allocated, or because the start bit itself (bit 0) has been allocated, or if requesting the start bit, because input bits have been allocated. A user can block itself from allocation: i.e. it should FreePotgoBits the bits it has and re-AllocPotBits if it is trying to change an allocation involving the start bit.

#### INPUTS

bits - a description of the hardware bits that the application wishes to manipulate, loosely based on the register description itself:

START (bit 0) - set if you wish to use start (i.e. start the proportional controller counters) with the input ports you allocate (below). You must allocate all the DATxx ports you want to apply

START to in this same call, with the OUTxx bit clear.

DATLX (bit 8) - set if you wish to use the port associated with the left (0) controller, pin 5.

OUTLX (bit 9) - set if you promise to use the LX port in output mode only. The port is not set to output for you at this time -- this bit set indicates that you don't mind if STARTs are initiated at any time by others, since ports that are enabled for output are unaffected by START.

DATLY (bit 10) - as DATLX but for the left (0) controller, pin 9.

OUTLY (bit 11) - as OUTLX but for LY.

DATRX (bit 12) - the right (1) controller, pin 5.

OUTRX (bit 13) - OUT for RX.

DATRY (bit 14) - the right (1) controller, pin 9.

OUTRY (bit 15) - OUT for RY.

#### RESULTS

allocated - the START and DATxx bits of those requested that were granted. The OUTxx bits are don't cares.

## 1.3 potgo.resource/FreePotBits

#### NAME

FreePotBits -- Free allocated bits in the potgo register.

#### SYNOPSIS

```
FreePotBits(allocated)
           D0
```

```
void FreePotBits( UWORD );
```

#### FUNCTION

The FreePotBits routine frees previously allocated bits in the hardware potgo register that the application had allocated via AllocPotBits and no longer wishes to use. It accepts the return value from AllocPotBits as its argument.

## 1.4 potgo.resource/WritePotgo

#### NAME

WritePotgo -- Write to the hardware potgo register.

#### SYNOPSIS

```
WritePotgo(word, mask)
           D0      D1
```

```
void WritePotgo( UWORD, UWORD );
```

#### FUNCTION

The WritePotgo routine sets and clears bits in the hardware

potgo register. Only those bits specified by the mask are affected -- it is improper to set bits in the mask that you have not successfully allocated. The bits in the high byte are saved to be maintained when other users write to the potgo register. The START bit is not saved, it is written only explicitly as the result of a call to this routine with the START bit set: other users will not restart it.

#### INPUTS

word - the data to write to the hardware potgo register and save for further use, except the START bit, which is not saved.

mask - those bits in word that are to be written. Other bits may have been provided by previous calls to this routine, and default to zero.