Self Training Guide

DLP Module 06 – DLP Arithmetic

Introduction

The DLP language has a set of simple arithmetic functions to allow the programmer to carry out basic mathematical operations while working with variables in the DLP.

This document discusses the functions available for carrying out common arithmetic operations and introduces two system registers, the Analogue Accumulators.

Before reading this document, you should have read the previous module(s) and be comfortable with the concepts discussed within. This document also assumes that you be familiar with the Q90 configuration software, and have successfully installed the DLP IDE software.

Additional details on the syntax of all DLP commands can be found in the online help.

In this document any DLP commands are presented in BLUE TYPEFACE while all DLP system variables and IO registers are in RED TYPEFACE.

The .ASM file for any DLP shown in this document is available separately.

This module contains help on:

- The Analogue Accumulators: ANLACC1 and ANLACC2
- Addition (ADDR and ADDS)
- Subtraction (SUBR and SUBS)
- Multiplication (MULR and MULS)
- Division (DIVR and DIVS)
- Increments and Decrements (INCANL and DECANL)

QTech Data Systems Ltd
Unit 6 Amuri Park
404 Barbadoes Street
Christchurch
New Zealand

Phone: +64 3 3663713
Fax: +64 3 3652815
E-mail: sales@qtech.co.nz
Web www.qtech.co.nz

ANLACC1 and ANLACC2, the Analogue Accumulators.

While the logic accumulator is useful in carrying out operations in the DLP, often the programmer will require that result of a function in a DLP won't be a logic TRUE or FALSE. Some functions exist that trigger the RTU to do something specific, such as sending a COS back to the base station, or the result will be a number, as in the case of most arithmetic functions.

The RTU maintains two analogue system registers called the Analogue Accumulators, ANLACC1 and ANLACC2.

ANLACC1 IS used to store the primary result of arithmetic functions, while ANLACC2 is used to store (when appropriate) things like a remainder or an overflow. Details are given below.

Addition using ADDR or ADDS

As with the analogue comparison functions outlined in module 04, different commands must be used depending on whether you are adding a register to another register (ADDR) or to a hard-coded constant setpoint (ADDS). In either case, the result is placed in ANLACC1. If the result is greater than 65535, then 1 is placed in ANLACC2 to signify 65536 and the reset is placed in ANLACC1.

Examples:

1st Number	2nd Number	Actual Result	ANLACC1	ANLACC2
50,000	10,000	60,000	60,000	0
60,000	5535	65,535	65,535	0
60,000	5536	65,536	0	1
60,000	10,000	70,000	4464	1

Subtraction using SUBR or SUBS

As with the analogue comparison functions outlined in module 04, different commands must be used depending on whether you are subtracting a register from another register (SUBR) or subtracting a hard-coded constant setpoint (SUBS). In either case, the result is placed in ANLACC1. If the result is less than 0, then 0 is placed in both ANLACC1 and ANLACC2.

Examples:

1st Number	2nd Number	Actual Result	ANLACC1	ANLACC2
50,000	10,000	40,000	40,000	0
100	100	0	0	0
100	200	-100	0	0

Multiplication using MULR or MULS

As with the analogue comparison functions outlined in module 04, different commands must be used depending on whether you are adding a register to another register (MULR) or to a hard-coded constant setpoint (MULS). In either case, the result is placed in ANLACC1. If the result is greater than 65535, then the result is calculated as a 32-bit unsigned number where ANLACC2 is the most significant word.
In other words the result can be found by (ANLACC2 *65536 + ANLACC1)

Examples:

1st Number	2nd Number	Actual Result	ANLACC1	ANLACC2
30,000	2	60,000	60,000	0
60,000	2	120,000	54464	1
60,000	20	1,200,000	20352	18

Division using DIVR or DIVS

As with the analogue comparison functions outlined in module 04, different commands must be used depending on whether you are dividing a register by another register (DIVR) or by a hard-coded constant setpoint (DIVS). In either case, the result is placed in ANLACC1. If the first number is not evenly divisible by the second number, the remainder is placed in ANLACC2.

Examples:

1st Number	2nd Number	Actual Result	ANLACC1	ANLACC2
10	2	5	5	0
15	2	7.5	7	1
33	7	4.7142	4	5

Single Increments and Decrements of Analogues using INCANL or DECANL

Frequently we simply want to increment or decrement an analogue register in response to a certain condition. To do so using ADDS or SUBS would be a little cumbersome as we would first have to add a "1" to the register and then as a separate operation, then use CPANL to copy the result from ANLACC1back into the original register.

The good news is that we can use INCANL or DECANL to increment a nominated analogue register and not have to worry about copying analog accumulators.

If a register is already at 65535 and the DLP tries to increment it using INCANL, the value will stay at 65535.

If the register is at 0 and the DLP tries to decrement it using DECANL, the value will remain at 0.