QTECH DATA SYSTEMS

Self Training Guide

DLP Module 01 – DLP Principles

Introduction

To be able to successfully write your own DLP's, it is first important to understand the principles behind which the DLP execution operates. This document describes these basic principles to provide a good foundation on which to begin your programming.

Before reading this document, you should 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:

- PROGINIT, PROGSTART and PROGEND
- The EQU statement.
- TELINP and TELOUT
- INITDIG and INITANL
- DLP Variable Types
- DLP Execution and the Logic Accumulator

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

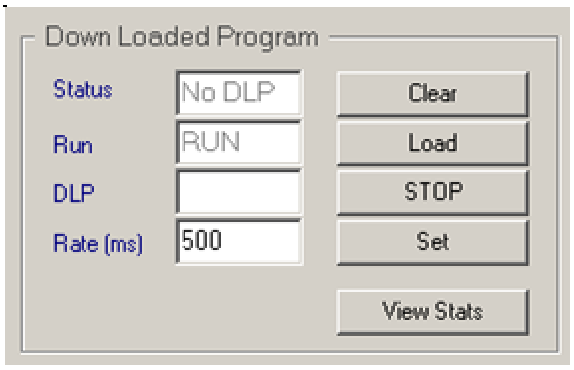
DLP Sections – PROGINIT, PROGSTART and PROGEND

Any DLP is made up of two sections. The first is the Initialization (or INIT) section, which is only ever executed once at start-up (Start-up can be when the power is turned on, when the reset button is pressed, or when a new DLP is loaded into memory). All of the initialization commands lie between the PROGINIT and PROGSTART statements.

The second section is the main body of the DLP that lies between the PROGSTART and PROGEND statements. This part of the DLP is executed sequentially the whole time the RTU is turned on. The frequency with which the DLP is executed is set in Q90.

The default rate is once every 500 ms but can be set between 10 ms and 2500 ms.

While it is tempting to set the DLP execution rate as low as possible to get things happening quickly, it is important to remember that executing the DLP uses significant RTU system resources that would otherwise be available for communications, data logging and servicing the IO. As such it is advisable to only make the DLP execute as frequently as it needs to to do the job required.


Down Loaded Program

Status: No DLP
Run: RUN
DLP:
Rate (ms): 500

Figure 1 - Q90 DLP Settings

If the DLP execution rate is set so small that the DLP has not finished executing before the next iteration is due to start, then the RTU will crash.
The only way to recover from this is to do a factory default reset of the RTU, so use caution!

EQU

In the INIT section you would usually make extensive use of the EQU (Equate) statement. This is a compiler directive that allows you to define names for various system registers to make the DLP more readable. For example, consider a system where have a “Pump 1 Fault” input wired into Digital Input 1 on the RTU. The native name for this register is RDIN1. We can use the EQU statement to tell DLP IDE that we would rather call it “rdi_Pump1_Fault”. This means we don’t need to have a table of all the I/O assignments sitting next to us as we write the DLP. You can imagine how difficult it would become if we had 150 digital inputs, and we had to always refer to them by their native names.

It is recommended good practice to equate every piece of I/O that the RTU has, even if it is not being used in the DLP.
This provides a fundamental level of documentation for the system so that it can be referred to in the future

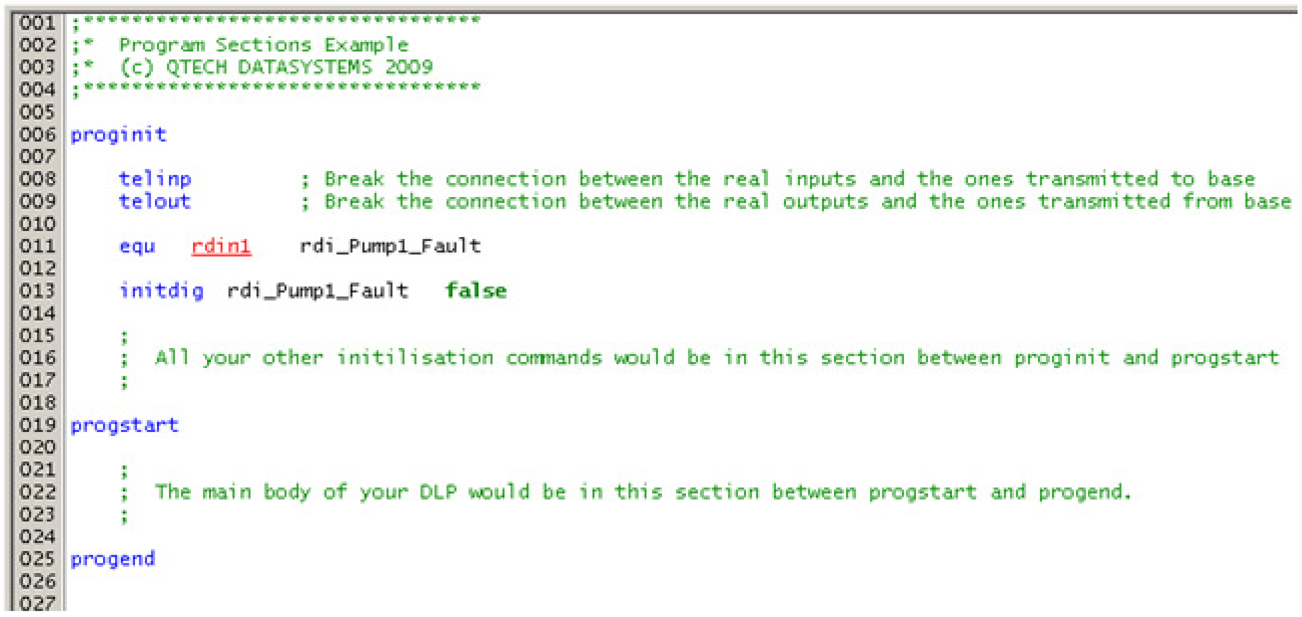
TELINP / TELOUT

Normally, the RTU will directly copy the I/O values that are presented at the input terminals back to the base station immediately. Likewise, any outputs that are sent from the base station are also presented at the outputs with no modification.

Often, the programmer may wish to modify the I/O before it gets to its final destination. For example, a programmer may wish to delay a high level alarm wired to a float for 15 seconds before transmission back to the base station so that short duration events are not reported.
The TELINP and TELOUT commands are used to break the automatic link between the real I/O terminals and the base station so that the DLP can modify the I/O as necessary. (See "DLP Variable Types" for more information)

INITDIG / INITANL

These commands stand for "Initialize Digital" and "Initialize Analogue" respectively. These commands are used anywhere in the DLP to force any register to a known value. As with other programming languages it is good programming practice to initialize all variables to a known state in the INIT section of the DLP.



```
;**********************************
;*  Program Sections Example
;*  (c) QTECH DATASYSTEMS 2009
;**********************************

proginit

    telinp          ; Break the connection between the real inputs and the ones transmitted to base
    telout          ; Break the connection between the real outputs and the ones transmitted from base

    equu   rdin1     rdi_Pump1_Fault

    initdig  rdi_Pump1_Fault    false

    ;
    ;  All your other initilisation commands would be in this section between proginit and progstart
    ;

progstart

    ;
    ;  The main body of your DLP would be in this section between progstart and progend.
    ;

progend

```

Figure 2 – PROGINIT example.

DLP Variable Types

Real Inputs (Analogue and Digital).
Examples: **RDIN12**, **RAIN6**
These registers represent the values that are physically present on the RTU Input terminals. The values are read by the RTU once every 100 ms, but do not update while the DLP is executing.

If there is no DLP running in the RTU, or the DLP does not have the TELINP and TELOUT commands then these values will be automatically copied to the corresponding Telemetry Input. (See Below).

Real Outputs (Analogue and Digital).
Examples: **RDOUT24**, **RAOUT3**
These registers represent the values that are being physically written to the RTU Output terminals. These registers are updated as the DLP execution proceeds, but are not physically published to the real world terminals until the end of the DLP iteration.

If there is no DLP running in the RTU, or the DLP does not have the TELINP and TELOUT commands then these values will be automatically copied from the corresponding Telemetry Output. (See Below)

Telemetry Inputs (Analogue and Digital).
Examples: **TDIN12**, **TAIN6**
These registers are the versions of the physical inputs that are transmitted to the base station, after being potentially modified by the DLP. At the base station they will be represented as RAIs or RDIs.

It is the TELEMETRY inputs that trigger such actions as COS messages, SMS messages (on SMS RTUs) and event logging triggers.

The DLP may write to these registers, but if there is no DLP running in the RTU, or the DLP does not have the TELINP and TELOUT commands then these values will be automatically copied from the corresponding Real Input at the end of each DLP iteration.

Telemetry Outputs (Analogue and Digital).
Examples: **TDOUT24**, **TAOUT3**
These registers represent the values that are being transmitted from the base station as RDOs and RAOs.

If there is no DLP running in the RTU, or the DLP does not have the TELINP and TELOUT commands then these values will be automatically copied to the corresponding Real Output.

Notional IO (Both inputs and outputs, analogue and digital).
Examples: **NDIN23**, **NDOUT2, NAOUT14, NAIN16**
Notional Registers are passed between the base station and the RTU and are intended for passing values that have no corresponding real world termination.
Examples:

- A Notional Analog Output would be used to transmit a setpoint from the base for the DLP to use to generate an alarm if RAIN3 gets above a certain value.
- A Notional Digital Input would be used to transmit the status of the high level alarm in the previous example back to the base station.
- A Notional Analog Input could be used to keep a running total of how many times in a day the high level alarm had been activated, or the total time the site spent with the alarm in the active state for the whole day.
- A Notional Digital Output could be used as a command sent from the base station reset the alarm, inhibit the alarm or to reset the fault counter.

Spare Registers (Analogue and Digital).
Examples: **SPA5**, **SPD14**
Spare registers are used internally in the DLP to hold values and as temporary registers for performing complex calculations or logic sequences. They are not transmitted to the base station and are not visible from Q90 either.
Examples:

- If your DLP needs an incremental timer to trigger certain actions, a SPA will most likely be used.
- If your DLP has flags that are set in one part of the DLP to show another part that certain processes should or should not happen, a SPD will often be used.
- If the DLP needs to "remember" what a certain value is during this iteration so that it can be used in the next iteration, then a SPA or SPD will be used as appropriate.

DLP Execution and the Logic Accumulator

The key to understanding the way that DLPs work is the "Logic Accumulator". This is a system register that stores the true/false result of the previous logic operation. The state of the Logic Accumulator affects the behaviour of almost every command in the DLP.

For Example, consider this small section from a DLP:

```
begin
    tce       rdin1
    setdig    rdout1
begin
    tce       rdin2
    resdig    rdout1
```

This DLP does two things, each with their own BEGIN statement signalling the start of a new "logic rung". This BEGIN statement resets the state of the logic accumulator to "True".

The TCE statement stands for "test if contact is energised", in other words, "Is the nominated input turned on?" The accumulator has to be true for the result of TCE to also be true. So more specifically, TCE means "Is the accumulator true AND is the nominated input turned on?" The result of this check is then written to the logic accumulator.
The SETDIG and RESDIG commands can be expressed in English as "If the logic accumulator is true, set (or reset) the nominated digital output".

Using what we have learned above, we can break this DLP down as follows:

BEGIN:			Set the logic accumulator to TRUE.
	TCE	RDIN1	Test If logic accum is true AND Digital input 1 is on. Write the result of the test into the logic accumulator.
	SETDIG	RDOUT1	If the logic accum is true, turn on digital output 1, else do nothing.
BEGIN:			Set the logic accumulator to TRUE.
	TCE	RDIN2	Test if logic accum is true AND Digital input 2 is on. Write the result of the test into the logic accumulator.
	RESDIG	RDOUT1	If the logic accum is true, turn off digital output 1, else do nothing.

In terms of functionality, this will turn on Digital Output 1 when Digital Input 1 is on, and will turn Digital Output 1 off when Digital Input 2 is on.

If both inputs are on at the same time, then the output will be turned off as the last section of the DLP will always take precedence when there is contention over a specific output.

In a RTU without a DLP all real inputs are connected to the telemetry inputs and all telemetry outputs are connected to the real outputs.
There is no manipulation of signal either way and so the RTU firmware copies the real signal directly to telemetry signals