

DEEP SEA ELECTRONICS DSEM812 Qt Manual

Document Number: 057-319

Author: Anthony Manton



057-317 ISSUE:1



Deep Sea Electronics Ltd Highfield House Hunmanby North Yorkshire YO14 0PH ENGLAND

Sales Tel: +44 (0) 1723 890099

E-mail: sales@deepseaelectronics.com Website: www.deepseaelectronics.com

DSEM812 Qt Manual

© Deep Sea Electronics Ltd.

All rights reserved. No part of this publication may be reproduced in any material form (including photocopying or storing in any medium by electronic means or other) without the written permission of the copyright holder except in accordance with the provisions of the Copyright, Designs and Patents Act 1988.

Applications for the copyright holder's written permission to reproduce any part of this publication must be addressed to Deep Sea Electronics Ltd at the address above.

The DSE logo and the name DSEControl® are UK registered trademarks of Deep Sea Electronics Ltd.

Any reference to trademarked product names used within this publication is owned by their respective companies.

Deep Sea Electronics Ltd reserves the right to change the contents of this document without prior notice.

Revision History

Issue No.	Comments
1	First Release

TABLE OF CONTENTS

Section

Page

1	Introduction	ED. 5 6 6 7 7
2	QT, C++ AND QML 2.1 PROPERTIES, SIGNALS AND SLOTS 2.1.1 SIGNALS. 2.1.1 PROPERTY DEFINITION 2.1.2 SIGNAL DEFINITION 2.1.3 SIGNAL USAGE. 2.1.2 SLOTS 2.1.2.1 PROPERTY DEFINITION 2.1.2.2 SLOT DEFINITION 2.1.2.3 SLOT METHOD	 8 9 9 9 9 9 9 10 . 10 . 10 . 11
3	INSTALLING THE QT ENVIRONMENT 3.1 PACKAGE INSTALL 3.1.1 CREATING THE MACHINE 3.1.2 CREATING A SHARED FOLDER 3.1.3 INSTALLING THE PACKAGE 3.1.4 CHECKING THE KIT	12 12 12 14 14 16
4	CONNECTING TO QT 4.1 FILE SYSTEM PATH 4.2 CHECKING CONNECTION TO THE DEVICE 4.3 START NEW PROJECT 4.4 I/O 4.5 CAN 4.5.1 PROCEDURE TO CONNECT TO THE CAN. 4.6 GPS 4.7 DEPLOYING THE APPLICATION TO THE DEVICE 4.7.1 ERROR CHECKING 4.7.2 DEPLOY APPLICATION 4.8 SELECTING AN APPLICATION TO AUTO RUN 4.9 CEASING (KILLING) A RUNNING APPLICATION	21 21 23 27 27 27 28 28 28 28 28 29 30 33
5	MAINTENANCE AND WARRANTY	34
6	DISPOSAL	34 . 34
7	MISC	34

1 INTRODUCTION

This document details the operation and setup requirements of the DSEM812 Qt Controller and Display, part of the DSEControl[®] range of products.

DSEM812 CODESYS variants are not covered in this document and are detailed within DSE *Publication 057-318 DSEM812 CODESYS Manual.*

Hardware, Specifications, Settings Pages and Installation Notes for all DSEM812 variants are detailed within DSE Publication 057-317 DSEM812 Operator Manual.

Knowledge of Linux and Qt (QML, C++ and JavaScript) is essential and assumed. This manual instructs the Qt programmer how to use Qt in conjunction with the DSE device. This manual does not give instruction for Linux, Qt, QML, C++ or JavaScript.

The manual forms part of the product and should be kept for the entire life of the product. If the product is passed or supplied to another party, ensure that this document is passed to them for reference purposes.

This is not a *controlled document*. DSE do not automatically inform on updates. Any future updates of this document are included on the DSE website at www.deepseaelectronics.com

Observe the operating instructions. Non-observance of the instructions, operation not in accordance with use as prescribed below, wrong installation or incorrect handling seriously affects the safety of the product, operators and machinery.

A robust metal case designed for chassis mounting houses the module. Connections are via locking plug and sockets.

The controller is supplied with no application program. The equipment manufacturer is responsible for creating and managing the application program and installing it in the controller. This is achieved using Qt programming. Contact DSE Technical Support for further details.

Qt is an application development framework for Linux Embedded systems.

Qt is not a programming language on its own. It is a framework written in C++ using signals and slots to pass information to/from Qt Modelling Language files (QML).



[]		()
		()
		()
[]		()
\Box		()
		()

1.1 CLARIFICATION OF NOTATION

Clarification of notation used within this publication.

Highlights an essential element of a procedure to ensure correctness.
Indicates a procedure or practice, which, if not strictly observed, could result in damage or destruction of equipment.
Indicates a procedure or practice, which could result in injury to personnel or loss of life if not followed correctly.

1.2 GLOSSARY OF TERMS

Term	Description
Application	The application is the program that allows the DSEM812 to control the
	machine it is connected to.
	The Application within the DSEM812 is designed and provided by the
	manufacturer of the complete machine.
Bootloader	The Bootloader is the program within the DSEM812 responsible for loading
	the Operating System.
C++	Programming language used alongside QML to create the complete
	application program.
CAN	Control Area Network. A high-speed data transmission system used
	extensively within the Automotive and Off-Highway industries.
Deploy	The compiled application is 'deployed' to the device folder /home/m812
ECU	Electronic Control Unit. For example, the DSEM812 device.
Firmware	The Firmware of the DSEM812 is the Operating System of the DSEM812
	that reads and executes the Application program.
FSD	Full Scale Deflection. For example, 0 mA to 20 mA is the Full Scale
	Deflection of a current sink input.
1/0	Input / Output. For example, "The I/O is taken out to an external terminal
1/0	strip in the user panel".
	Integrated Development Environment. For example, the Qt application that
IDE	runs on the host PC is an IDE, containing code editors, compilers and
	much more.
book	An Input, where x is the connector and yyy is the input number. For
тхууу	example, IB003 means Input 3 on Connector B.
Nano	An editor to allow text files to be altered.
PLC	Programmable Logic Controller. Industrial computer used primarily for the
	automation of electromechanical machinery.
PWM	A digital signal is used to represent an analogue value by using Pulse
PWMi	Width Modulation. The mark-space ratio of a square wave changes to
	represent the value.
	Used for many control applications including proportional valves.
	PWM= Voltage control.
	PWMi = Current control.
Off-Highway	An industrial vehicle used primarily "off road". For example construction
	and farm machinery. A wider interpretation includes on road access
	platforms, emergency vehicles and other industrial machinery, used either
	on the road, or off road.
Pin	A male or female pin connection in a housing (plug or socket).

QML	Qt Modelling Language. Used to design/describe the display elements on
	the DSEM812 Qt variant.
Qt	Application development system supported by DSEM812 Qt Variant
	Pronounced 'Cute'.
Qxyyy	An Output, where x is the connector and yyy is the output number. For
	example QB002 means Output 2 on Connector B.
Remote Shell	A <i>terminal</i> to allow commands to be run on the target device.
Run.sh	Shell program executed at device boot time. Among other functions, this
	file instructs the device which application to execute at boot of the device.

1.3 RELATED INFORMATION

This document refers to and is referred by the following DSE publications which are obtained from the DSE website: <u>www.deepseaelectronics.com</u> or by contacting DSE technical support: <u>support@</u> <u>deepseaelectronics.com</u>.

1.3.1 TECHNICAL INFORMATION

DSE Part	Description
055-267	DSEM812 Datasheet
057-317	DSEM812 Operator Manual

1.4 SAFETY INSTRUCTIONS

1.4.1 GENERAL

- These instructions are for authorised persons according to the EMC and low-voltage directives. The device must be installed, connected and put into operation by a qualified electrician.
- It is not permissible to open the controller or to modify or repair the controller. Modification or repairs to the wiring could result in dangerous malfunctions. Repairs to the controller must be performed by DSE. Contact your original equipment supplier in the case of malfunction.
- When the device is unpowered, ensure that no connection pins are connected to a voltage source. Thus, when the supply is switched off, the supply for the electronics, the power outputs and the external sensor supply must be switched off together.
- The controller heatsink at the rear heats up beyond normal ambient temperature during operation. To avoid danger caused by high temperatures, protect against contact.
- The customer is responsible for performing risk analysis of the mobile working machine and determining the possible safety related functions. The user is responsible for the safe function of the application programs created. If necessary, they must additionally carry out an approval test by corresponding supervisory and test organisations according to the national regulations.
- All connectors must be unplugged from the electronics during electrical welding and painting operations.

1.4.2 INSTALLATION NOTES

- Follow the instructions of the connector manufacturer, specifically with respect to preventing water from entering the device. See Section entitled *Cables, Connectors, Harnesses and Spare Parts* for details of DSE Part Numbers.
- To maintain IP67 rating where connectors have unused pins, ensure the use of a suitable Blanking Insert. In the case of a completely unused connector, the plug must be inserted, fully populated with Pin Blanking Inserts. See Section entitled *Cables, Connectors, Harnesses and Spare Parts* for details.
- M12 protection plugs (supplied) must be installed in both the USB and Ethernet interfaces to ensure IP67 rating when the connectors are not in use. Tighten to 0.8 Nm (0.6 lbf ft). Where IP protection is required when the interfaces are in use, suitable O-rings must be fitted.
- The heatsink must be wired to vehicle ground to comply with EMC guidelines. A screw connection point is provided for this purpose. A metallic screw must be used to create an electrical connection to vehicle / machine ground.

2 QT, C++ AND QML

NOTE: Knowledge of Linux and Qt (QML, C++ and JavaScript) is essential and assumed. This manual instructs the Qt programmer how to use Qt in conjunction with the DSE device. This manual does not give instruction for Linux, Qt, QML, C++ or JavaScript.

Qt (pronounced 'Cute') is an application development system used by the Linux based DSEM812 devices. C++, QML and JavaScript are used to program the device.

QML is *Qt Modelling Language*, a mark-up language designed to aid development of user interfaces. QML elements support JavaScript both inline and via included .js files.

QML is used to describe only the user interface. Where interaction with device hardware is required, Integration with C++ is used.

Qt allows the use of externally programmed libraries to 'extend' the functionality of the system. This includes the use of DSE supplied libraries to allow for example, the use of the device I/O.

C++ and QML work together to provide the full functionality of Qt.



2.1 PROPERTIES, SIGNALS AND SLOTS

Properties, Signals and Slots are the mechanism that binds together C++ and QML. C++ classes derived from *QObject* or a subclass are registered to allow them to be created as objects within QML.

- Properties are values associated with the object. QML reads and/or writes from/to the properties. For example, a property of a class used to monitor the inputs of the device may be read by the QML to know the voltage applied to the input.
- Signals are emitted by the C++ class upon specific actions occurring within the class. Typically, a signal is emitted when a property value is changed in the C++ class. For example, if a device input monitored by the C++ class changes value, a signal is sent to inform the QML that the input has changed. *Signal Handlers* are used in the QML to act upon signals from the C++.
- A slot is a method of the class called when the property is changed by the QML. For example, if the QML changes a property linked to the state of a device output, the C++ class detects the change and calls the slot method that handles the *setting* of the class variable. In turn this may also trigger a signal as the linked property has also changed.

Examples of properties, signals and slots are given in the following subsections.

2.1.1 SIGNALS

QObject Signals are emitted from C++ and typically used to trigger callback functions in the QML JavaScript.

Signals are emitted by an object when it is required to inform about a change in the object. The *Emit* keyword is used to do this.

Good practice dictates that a signal is used to indicate any properties that have changed.

2.1.1.1 PROPERTY DEFINITION

Within the class *.h* file a Q_PROPERTY is defined. In this example the property is *read-only* by the QML due to the omission of a WRITE slot definition. For details of this, see section entitled *Slot Definition* elsewhere in this document.

	Name and type of the property as presented to QML	
Q_PROPERTY(quint32 raw	MEMBER mRaw	NOTIFY rawChanged)
Parameter	Description	
MEMBER	Defines the class variab property. Such class variable	le that is used to link to the QML object riables are prefixed 'm' to indicate <i>MEMBER</i> .
NOTIFY	Defines the signal emitted upon change of the property. The signal must be prototyped in the <i>.h</i> file but does not required an associated	

upon change of the property.

2.1.1.2 SIGNAL DEFINITION



Definition of the signal method within class *.h.* In this case the value of *raw* is passed as a parameter of the function.

method creating in the .cpp file. The signal is automatically emitted

2.1.1.3 SIGNAL USAGE

To receive a notification when a signal is emitted for an object, the object definition includes a signal handler named on <Signal>, where <Signal> is the name of the signal, with the first letter capitalised. The signal handler contains JavaScript code executed when the signal handler is triggered.



Definition of the signal handler within a *.qml* file In this case the value of *raw* is passed as a parameter of the function.

2.1.2 SLOTS

QObject Slots are C++ methods that allow them to be called upon the change of a property.

The class *.h* file defines the C++ class a derived from *QObject* and defines methods as *public slots* to make them accessible to QML.

2.1.2.1 PROPERTY DEFINITION

Within the class .h file a Q_PROPERTY is defined. In this example the property is read/Write.

		Name and type of property as presented to QML	of the ented	
Q_PROPERTY(bool	state	MEMBER mState	WRITE setState	NOTIFY stateChanged)

Parameter	Description
MEMBER	Defines the class variable that is used to link to the QML object
	property. Such class variables are prefixed 'm' to indicate member.
WRITE	Defines the name of a method called when QML changes the value of the property. This is used to control access to the class variable. Typical application of the slot method is to check if the property value has changed, validate the new property value, update the class member variable with the new property value and finally emit a signal to inform QML of the change.
NOTIFY	Defines the signal emitted upon change of the property. The signal must be prototyped in the <i>.h</i> file but does not required an associated method creating in the <i>.cpp</i> file. The signal is automatically emitted upon change of the property.

2.1.2.2 SLOT DEFINITION



Definition of the slot method within class *.h.* In this example a pointer to the property is passed into the slot method. This is used to update the class variable.

2.1.2.3 SLOT METHOD

In this example, a device output is controlled, based upon the value of the property pointed to in the method call.

- The output is set to the state requested by the QML property state.
- If the property value of *state* has changed (if it is not the same as the class variable *mState*) then the class variable is updated and the signal *stateChanged* is emitted, passing the updated value along with the signal.

```
void Outputs::setState(bool &state)
{
    dse_io_output_set_state(static_cast<dse_io_output_pin_t>(mOutput), state);
    if (state != mState) {
        mState = state;
        emit stateChanged(mState);
    }
}
```

3 INSTALLING THE QT ENVIRONMENT

NOTE: *DSEvm.zip* containing the Virtual Disk Image is available from support@deepseaelectronics.com.

Extract the VDI (Virtual Disk Image) from the containing zip file. The extracted image is over 6 GB. The VDI is designed to be used by Oracle VM VirtualBox PC software and provides a Ubuntu operating system with Qt 5.1 preinstalled and configured for use with DSEM812.

VirtualBox is available from https://www.virtualbox.org/wiki/Downloads

3.1 PACKAGE INSTALL

Install VirtualBox. A new machine needs to be created with the supplied virtual disk image.

3.1.1 CREATING THE MACH	line	First select New
Oracle VM VirtualBox Manager <u>File</u> <u>Machine</u> <u>H</u> elp		- 🗆 X
Tools 🖉 🗄	Preferences Import Export New (Ctrl+N)	
Windows Constraints in the second se	Welcome to VirtualBox! The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can import, add and create new VMs using corresponding toolbar buttons. You can popup a tools of currently selected element using corresponding element button. You can press the F1 key to get instant help, or visit www.virtualbox.org for more information and latest news.	

Then enter a name, select a folder and select Linux, Ubuntu (64-bit):

? \times

Create Virtual Machine

Name and operating system

Please choose a descriptive name and destination folder for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name:	Lorem Ipsum
Machine Folder:	C:\Users\Public\Documents ~
<u>Type</u> :	Linux 🔹 🛀
Version:	Ubuntu (64-bit)
	Expert Mode Next Cancel

Then select Next, use a minimum of 1GB of memory.

Select Next and choose "Use an existing virtual disk file" and select the supplied virtual disk image:

	? ×
← Create Virtual Machine	
Hard disk	
If you wish you can add a virtual hard disk to the new mach create a new hard disk file or select one from the list or from using the folder icon.	nine. You can either n another location
If you need a more complex storage set-up you can skip this changes to the machine settings once the machine is create	is step and make the ed.
The recommended size of the hard disk is 10.00 GB .	
O Do not add a virtual hard disk	
O Create a virtual hard disk now	
Use an existing virtual hard disk file	
Empty	▼
	Cł
Create	ce Cancel

Then select	Add:	Select Add
	Direm Ipsum - Hard Disk Selector ?	
	Medium	
	Add Refresh	
	Search By Name 🔻	<i>S</i>
	Choose Cance	el

Then select the supplied virtual disk image and select Create.

3.1.2 CREATING A SHARED FOLDER

Once the Virtual Machine has been created, a shared folder needs to be created to transfer the Qt package on to the virtual machine. First, select the machine and select Settings:



🛞 Lorem Ipsum - Sett	ings	?	×
General	Shared Folders		
System	Shared Eolders		
Display	Name Path Access Auto Mount Machine Folders	At	Ac
Storage			fo
Audio	Select Shared Folders		
Network			
Serial Ports			
🖉 USB	Then click Add		
Shared Folders			
User Interface			
	OK	Can	icel

Select the folder path to the supplied Qt package. Use "Shared" for the Folder Name:

😟 Edit Share	2	?	×
Folder Path:	C:\Users\Public	Downloads	~
Folder Name:	shared		
	Read-only		
	Auto-mount	When checke	d, the gu
Mount point:			
	OK	Cano	el

Start the Virtual Machine and ensure the shared folder is accessible. Start the machine, allowing it to boot to the desktop, then install Guest Additions by selecting "Install Guest Additions CD Image":



Wait for it to install, then reboot the virtual machine. Then, open a terminal (Ctrl+Alt+T) and type

sudo cp /home/user/shared/m812_qt_package.tar.gz /home/user/Desktop/; sudo chown user:user /home/user/Desktop/m812_qt_package.tar.gz

The password for the user account is *password*. The package is now available on the desktop to extract.

3.1.3 INSTALLING THE PACKAGE

Once the package is on the Desktop, it is ready to install. Double click it and extract it on the desktop. Open a terminal on the Desktop (Ctrl+Alt+T) and run the command

```
cd ~/Desktop; ./install.sh
```

If prompted for a password, enter *password*. The virtual machine is now ready to compile and program for the chosen device (DSEM812).

3.1.4 CHECKING THE KIT

Open Qt Creator:

Ubuntu Desktop		†4	En 💌	● 	3:36 PM	₩
m812_qt_pac :age. Qt Creator	m812_rsa_openssh					
Examples	sysroot.tar.gz					
Documentation						
install.sh						

Then select File, Open File or Project:



Choose an example from the Examples directory on the Desktop. Qt projects have the extension *.pro*. In this case, the Examples/Touchscreen/Touchscreen.pro was selected:

Qt Creat	or:		🏗 🖬 🖦 🗤 3:42 PM 🔆
	<u>F</u> ile <u>E</u> di	t <u>B</u> uild <u>D</u> ebug <u>A</u> nalyze <u>T</u> o	ols <u>W</u> indow <u>H</u> elp
\odot			😢 🗊 Open File
Qt	Welcome	Projects	O Recent •
	Edit Design	Examples Tutorials	□ Desktop □ main.cpp 375 bytes 13 Feb □ Documents □ main.qml 2.8 kB 13 Feb □ gmLqrc 87 bytes 13 Feb □ Downloads □ Touchscreen.pro 172 bytes 13 Feb
	Debug Projects Projects Help	New to Qt? Learn how to develop your own applications and explore Qt Creator. Get Started Now	Misic Pictures Videos shared VBox_G + Other Locatio
a		Qt Account Contine Community Blogs	All Files
	1 1	User Guide	Cancel Open
		P. Type to locate (Ctrl	1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 8 Test Results 🖨 🧰 🖬

Select M812 in the **Build & Run** section on the left:

Touchsc	reen - Qt (Creator			🏚 🖪 🔜 🕬	3:43 PM 🔱
Q	<u>F</u> ile <u>E</u> di	t <u>B</u> uild <u>D</u> ebug <u>A</u> nalyze <u>T</u> ools <u>W</u> ind	low <u>H</u> elp	, ,	_	
Ot	QL	Manage Kits	>	Build Settings		
			× .	Edit build configuration: Debug * Add * Remove Rename Clone		
	Edit	Active Project		General		
	24	Touchscreen *				
	Design	Import Existing Build		Shadow build: 🗸		
				Build directory: /home/user/Desktop/Examples/build-Touchscreen-M812-Debug	Browse	
	Debug	Build & Run		Puild Stops		
	Projects	Desktop		Build Steps		
	0	💻 M812		qmake: qmake Touchscreen.pro CONFIG+=debug CONFIG+=qml_debug	Details 👻	
	Help	Build		Make: make -j1 in /home/user/Desktop/Examples/build-Touchscreen-M812-Debug	Details -	
		Run		Add Build Step *		
!!!		Project Settings		Add Balld Step		
	T	Editor		Clean Steps		
A		Code Style		Make: make clean -i1 in /home/user/Desktop/Examples/build-Touchscreen-M812-Debug	Details -	
	Debug	Clang Code Model		Add Clean Stop x		
<u>a</u> ,		Clang Tools		Add clean step	Scanning QML II	mports
				Build Environment	Parsing QML	Files
				Use System Environment	Desidie - Desiret "Te	
	\sim			•	Reading Project To	uchscreen
		P. Type to locate (Ctrl	ues 2 Se	earch Results 3 Application 4 Compile Out 5 QML Debugg 6 General Mess 8	Test Results 🗧	-



Select the build button (hammer) in the bottom left:

If the build succeeds with no errors, you should have the following if opening the "Compile Output" window:

Touchsc	reen - Qt (Creator		tų.	En 📧 🜒) 3:45 PM 🔱
	<u>F</u> ile <u>E</u> di	t <u>B</u> uild <u>D</u> ebug	<u>Analyze</u> <u>To</u>	ols <u>W</u> indow <u>H</u> elp	
0		Projects	≑ ▼. ⇔ ⊞•	🖂 < > <no document=""> + ×</no>	8+
Qt	Welcome	Touchscree	20		
	Edit			Open a document	
	\sim			 File > Open File or Project (Ctrl+O) 	
	Design			 File > Recent Files 	
	Debug			Tools > Locate (Ctrl+K) and - type to open file from any open project - type cospares/cnatter are to iume to a class definition	
				- type m <space><pattern> to jump to a function definition - type f<space><filename> to open file from file system</filename></space></pattern></space>	
	Projects			- select one of the other filters for jumping to a location	
	🕜 Help			Drag and drop files here	
				Compile Output	
A	Touchscreen	Open Document	s ≑⊟+	Payroot*/opt/ast/mair/arm-out/arm-out/armous/mutaroot-innux-gnueabinf/sysroot -g -std*gnu+il -wait -w -ukcenikawi -hpit -out(-00T GUILB -00T GUILB -00T GWINK_LBB -00T CME LBB -1.7 Touchscreen - 1.71/opt/dse/mB12/arm-buildroot-li - arwohinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/arm-buildroot-linux-gnueabinf/sysroot/usr/include/at5/fotgui-it/out/dse/mB12/fotgui-it/out/dse/mB12/fotgui-it/out/dse/mB12/fotgui-it/fotgui-it/fotgui-it/fotgui-it/fotgui-it/fotgui-it/fotgui-it/fotgui-it/fotgui-it/fo	nux-gnueabihf/sysroot/ arm-buildroot-linux- OtOml -I/opt/dse/m812/
a,	Debug			arm-buildroot-linux-gnueabihf/sysroot/usr/include/qt5/QtNetwork -I/opt/dse/m812/arm-buildroot-linux-gnueabihf/ QtCore -II/opt/dse/m812/arm-buildroot-linux-gnueabihf/sysroot/usr/include -I/opt/dse/m812/mkspecs/devices/L grc_gml.ogrc_gml.opp	sysroot/usr/include/qt5/ inux-buildroot-g++ -o
				<pre>/opt/dse/m812/bin/arm-buildroot-linux-gnueabihf-g+sysroot=/opt/dse/m812/arm-buildroot-linux-gnueabihf/sysr main.o grc_gml.o</pre>	oot -o Touchscreen buildroot-linux-
				IS:45:02: The process "/usr/bin/make" exited normally. IS:45:02: Elapsed time: 00:15.	Build
		🔳 🔎 Type to l	ocate (Ctrl	1 Issues 2 Search Results 3 Application 4 Compile Out 5 QML Debugg 6 General Mess 8 Test	Results 🗢 🖃 🔳

This project is now ready to download. Select the Run icon at the bottom left:



If the program starts on the M812, the kit is correct. This is assuming the M812 is connected to the same network as the virtual machine and the M812 has the IP address of 192.168.1.100.

If the M812 has a different IP address, see section entitled *Checking Connection to the Device* elsewhere in this document.

4 CONNECTING TO QT

NOTE: Prepare the Qt installation before following this section. See previous section entitled *Installing the Qt Environment* elsewhere in this document.

NOTE: Knowledge of Linux and Qt (QML, C++ and JavaScript) is essential and assumed. This manual instructs the Qt programmer how to use Qt in conjunction with the DSE device. This manual does not give instruction for Linux, Qt, QML, C++ or JavaScript. Comprehensive online documentation for Qt is provided at <u>https://doc.qt.io/qt-5/</u>

DSEM812 Qt variant communicates with, and is programmed by, the Qt Integrated Development Environment (IDE). Online documentation for the IDE is available at <u>https://doc.qt.io/qtcreator/</u>

4.1 FILE SYSTEM PATH

Path	Description
/home/m812	Working folder for all user operations.
	Contains Run.sh (boot time setup) and all applications that have been
	deployed to the device.

4.2 CHECKING CONNECTION TO THE DEVICE

First use the device *Settings Pages* to configure the ethernet port as required. Full instructions how to enter and utilise the *Settings Pages* are contained in *DSE Publication 057-317 DSEM812 Operator Manual.*





Opcions			
ilter	Devices		
Kits	Devices Android QNX SSH		
- Environment	Device: M812 (default for Generic Linux)	+-	
Text Editor	General	The	n select Devices
K FakeVim	Name: M912		
🛛 Help	Type: Generic Linux	Test	
() C++	Auto-detected: No	Show Running Processes.	
A Qt Quick	Current state: Unknown	Deploy Public Key	
Build & Run	Type Specific	Open Remote Shell	
Debugger	Machine type: Physical Device		
Designer	Authentication type: Default Specific key		
E Analyzer	Host name: 192.168.1.28 SSH port: 22 Check host key		
Version Control	Free ports: 10000-10100 Timeout: 10s 🗘		
Devices	Username: m812		
Code Pasting	Private key file: /m812_rsa_openssh Browse Create New		
🖉 Language Client	GDB server executable: Leave empty to lo		
▲ Testing			

ANOTE: Changing settings other than those detailed may cause unexpected errors.

	Devices Android QNX SSH		Ensure M812 is selec	ted
	Device: M812 (default for Generic Linux)			
	General			
	Name: M812 Type: Generic Linux Auto-detected: No Current state: Unknown		And enter the IP addr of the device on your network.	ess
	Type Specific			
	Machine type: Physical Device Authentication type: Default ● Specific key Host name: 192.168.1.28 SSH port: 22 ‡ ♥ Check host Free ports: 10000-10100 Username: m812 Private key file: /m812_rsa_openssh GDB server executable: Leave empty to lo	it key		
Test to ch	eck the settings and test the			
ections to t	he device.		<u>A</u> dd	
	1002		Remove Set As Default	
	Name: M812 Type: Generic Linux Auto-detected: No Current state: Current state: Unknown Type Specific Machine type: Physical Device Authentication type: Default Machine type: Default Specific. Default Machine type: Default Specific key Host name: 192.168.1.28 SSH port: Yree ports: 10000-10100 Timeout: 10s Username: m812 Private key file: /m812_rsa_openssh Browse Create New GDB server executable: Leave empty to lo		Set As Default Test Show Running Processes Deploy Public Key Open Remote Shell	
	Name: MB12 Type: Ceneric Linux Auto-detected: No Durrent state: Current state: Unknown Type Specific Machine type: Machine type: Default © Specific key Host name: 192.168.1.28 Spst name: 192.168.1.28 Gob name: 192.168.1.28 Private key file: //m812_rss_opensch Browse Create New GOB server executable: Leave empty to lo		Set As Default Test Show Running Processes Deploy Public Key Open Remote Shell	
	Name: M812 Type: Ceneric Linux Auto-detected: No Current state: Current state: Unknown Type Specific Machine type: Machine type: Default • Specific key Host name: 192.168.128 SSH port: 22 Ust name: 192.168.128 SSH port: 22 GDB struer executable: Leave empty to lo Connecting to host Create New Checking kernel version Linux 4.9.88_2.0.0-00018-g8f4019a armv7l Checking if specified ports are available All specified ports are available	The re Click	Set As Default Show Running Processes Deploy Public Key Open Remote Shell esults of a successful te <i>Close</i> to continue	est.
	Name: MB12 Type: Ceneric Linux Auto-detected: No Current state: Current state: Unknown Type Specific Machine type: Machine type: Default	The re Click	Set AS Default Show Running Processes Deploy Public Key Open Remote Shell esults of a successful te <i>Close</i> to continue	est.
	Name: MB12 Type: Ceneric Linux Auto-detected: No Current stat: Unknown Type Specific Machine type: Mathentication type: Default • Specific key Host name: 192.168.1.28 Gost name: 192.168.1.28 Private key file: //m812_rsa_openssh Beserver executable: Leave empty to lo Connecting to host Create New CbB server executable: Leave empty to lo Connecting to host Checking kernel version Linux 4.9.88_2.0.0-00018-g8f4019a armv7l Checking if specified ports are available All specified ports are available. Checking whether an SFTP connection can be set up SFTP service available. Checking whether rsync works Checking whether rsync works rsync is functional.	The re Click	Set As Default Show Running Processes Deploy Public Key Open Remote Shell essults of a successful te <i>Close</i> to continue	est.

4.3 START NEW PROJECT

To begin, start a new project as shown.



😣 💷 Ot Quick Appli	cation - Empty
	Define Project Details
Location Build System > Details Kits Summary	Minimal required Qt version: Qt 5.11
nsure <i>Qt 5.11</i> is selected. The resion supported by the device	nis is the Qt ce.
	< <u>Back</u> <u>Next</u> > Cancel
Control Contro	Cation - Empty Kit Selection The following kits can be used for project MyNewProject: Type to filter kits by name Cy Select all kits
Summary	
	Ensure <i>M812</i> is selected. If this is not correct, click <i>Back</i> and recheck <i>Minimal Qt Version</i> is set to <i>Qt 5.11</i> .
🔵 💷 Ot Ouick Appli	< <u>Back</u> <u>Next></u> Cancel
Location Build System Details Kits	Project Management Add as a subproject to project: <none> Add to version control:</none>
Summery	Summary of the project location and files to be created.

The new project is created with a simple Window.



However, we will modify this to provide a simple application to show something on the screen.



Deploy the application as described in the section entitled *Deploying the Application to the Device* elsewhere in this document.

Results of the application running on the device:

0	My New Application	\Box
\Box		
\Box		()
()		()
()		()
()		()
()		()
()		()

4.4 I/O

Device I/O is controlled and read using <i>dseio</i> library. Add the library to <i>LIBS</i> within the <i>.pro</i> project file.	<pre># Add the required DSE libraries LIBS += -ldseio</pre>
Include to any <i>.h</i> or <i>.cpp</i> files that use the library.	<pre>#include "dseio.h"</pre>
Then initialised the library before use within main.cpp.	<pre>// Initialise the dseio library dse_io_lib_init();</pre>

NOTE: For full details, see *Examples\Inputs* and *Examples\Outputs* provided in the DSEM812 Qt package.

4.5 CAN

DSEM812 is equipped with three CAN ports. Support for configuring them is provided using *Idsenetwork* library.

Add the library to <i>LIBS</i> within the . <i>pro</i> project file.	<pre># Add the required DSE libraries LIBS += -ldsenetwork</pre>
Within the cpp file include the required header files	<pre>#include <qcanbus> #include "can.h" #include "dsenetwork.h"</qcanbus></pre>
Example CAN port configuration 250 = 250 kbit/s 0=Interface 0 (CAN1)	<pre>dse_network_can_config_t config; config.size = sizeof(dse_network_can_config_t); config.bitrate = 250; config.id = 0; dse_network_set_can_config(m_interface, &config);</pre>

4.5.1 PROCEDURE TO CONNECT TO THE CAN

NOTE: A CAN application that does not *receive* CAN messages (used for *transmit* only) must still implement the *FramesReceived* method. This provides a mechanism to retrieve frames from the QCANBus device and prevents the receive memory overflowing.

ONOTE: For full details, see *Examples**CAN* provided in the DSEM812 Qt package.

- Use *dse_network_set_can_config* to configure the device hardware.
- Create the QCANBus SocketCAN.
- Connect the SocketCAN to the device hardware,
- Connect QCANBus device signal *FramesReceived* to the event handler within your own cpp file.
- The event handler you provide for *FramesReceived* is used to filter and parse the incoming CAN frames (messages) as required.

4.6 GPS

NOTE: A suitable external GPS antenna is required. For full specification see DSE Publication *057-317 DSEM812 Operator Manual.*

Within the QML file, the following imports are required:

```
import QtPositioning 5.11
import QtLocation 5.11
```

QML PositionSource item is used to retrieve the location at the given updateInterval (ms).

```
// Get the GPS location
PositionSource {
    id: gpsLoc
    updateInterval: 1000
    active: true
    onPositionChanged: {
        // Code to handle the change in position if required
    }
}
```

4.7 DEPLOYING THE APPLICATION TO THE DEVICE

ANOTE: Successful communications with the device is required before deploying the application. Refer to section entitled *Checking Connection to the Device* elsewhere in this document.

4.7.1 ERROR CHECKING



4.7.2 DEPLOY APPLICATION

NOTE: Ensure *Device Settings* application is closed (exited) before sending your application to the device. Failure to do this may result in multiple applications displaying items on the display at the same time.

NOTE: When updating an application on the device, ensure the 'old' version running on the device is closed before writing the updated version to the device. For further details, see section entitled *Ceasing (killing) a Running Application* elsewhere in this document.

NOTE: The device accepts multiple applications to be installed to it at the same time. To select the application to start at device boot up, refer to section entitled *Selecting an Application to Auto Run* elsewhere in this document.

NOTE: Where multiple applications are running on the device simultaneously, ensure there is no conflict in the use of the display. It is recommended that only one application utilises the display.



4.8 SELECTING AN APPLICATION TO AUTO RUN

NOTE: The device detects the presence of *Run.sh* and creates a new 'factory set' file if not present.

To configure the device to automatically start an application upon device boot up we must edit *Run.sh* on the device.

Options are

 Maintain a Run.sh file in the project folder. Deploy this file by adding the following to the project file (.pro file)

```
scripts.files += $$PWD/run.sh
scripts.path = .
INSTALLS += scripts
```

This method is detailed within the Examples provided with the DSEM812 Virtual Machine Package.

• Modify Run.sh in the location /home/m812 as detailed below :





Press CTRL X when done.



Close the Terminal window and power cycle the device to check the change. If your application fails to start at boot up:

- Use *ls* to check the presence and exact spelling of the application on the device in folder */home/m812.* (Check case)
- Use Nano to check the path is correctly entered (case sensitive).
- Check the application by executing it directly from the *Remote Terminal*. Enter its name at the command prompt.

4.9 CEASING (KILLING) A RUNNING APPLICATION

To cease a running application, use a Remote Shell:



5 MAINTENANCE AND WARRANTY

The device is *Fit and Forget*. As such, there are no user serviceable parts within the controller. In the case of malfunction, you should contact your original equipment manufacturer (OEM).

DSE Provides limited warranty to the equipment purchaser at the point of sale. For full details of any applicable warranty, refer to the original equipment supplier (OEM).

6 DISPOSAL

6.1 WEEE (WASTE ELECTRICAL AND ELECTRONIC EQUIPMENT)

If you use electrical and electronic equipment you must store, collect, treat, recycle, and dispose of WEEE separately from your other waste



7 MISCELLANEOUS

This product includes copyrighted third-party software licensed under the terms of the GNU General Public License. A copy of the corresponding source code for all included third-party software is available on request, please contact DSE Technical Support for additional information.

This Page is Intentionally Blank

This Page is Intentionally Blank