Zoltán Kiss - Endrich Bauelemente Vertriebs GmbH

Developing human machine interface using smart display modules part 2.

The previous article we detailed the properties of DLOGIC's Smart Display Modules and the possibility of designing a Human Machine Interface. In case of using conventional TFT touch displays the interface design, hardware and software debugging consumes a lot of time, better to take in consideration using a ready, economic, professionally designed smart display module and limit the HMI design to pure software development to minimize human resources and time-to-market of the finished product. After reviewing the advantages of this technology, it is time now to see how easy to implement software HMI by using a cross-platform development tool called Qt on Linux.

Cross platform technology in SDM programming

To make the first prototype of the HMI, we need to buy a development kit, which contains all the components required next to the display module for the development process. We will find in the package a mounting frame, cables, connectors and a PCB offering physical connection to the communication interfaces of the SDM.

According to the sales philosophy of the manufacturer, the product purchased for mass production comes without any cables, accessories, even no power supply is provided to keep the price level as low as possible.

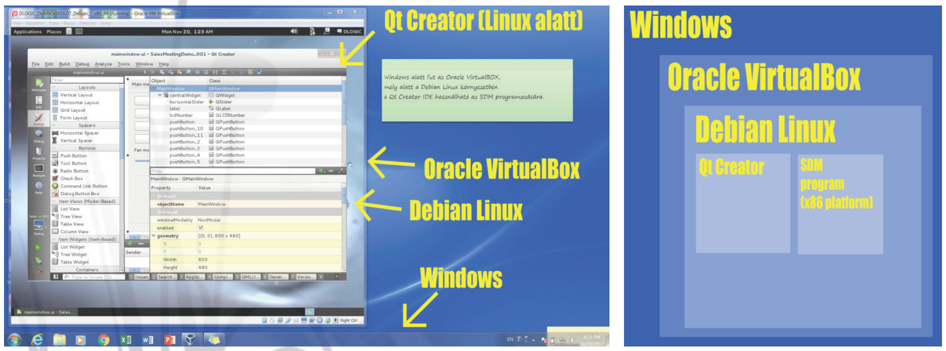
It is supposed that the required unique mounting is already solved, tailor made connections for the application requirements are already created in the final product. The possibility of supplying the SDM with any DC voltages between 9-36V offers more advantage s and flexibility than to provide a fixed 230V AC to 12V DC power supply.

When a development kit is purchased the software support is a part of the deal after providing the unique MAC address of the ethernet port of the device to DLOGIC.

1|DLogic development kit for cross platform HMI design

A ready-made virtual computer Debian Linux image, with all software component preinstalled on it will arrive including the Qt Creator cross platform development IDE. The only thing is to install one of the available virtual machine programs such as ORACLE's VirtualBox on the Windows PC and import the image. A Debian Linux distribution provides good development environment and stability. The essence of the cross-compiling theory is, that the developer uses a computer for writing and debugging the HMI software, and later deploys the executable for a different platform that the compiler itself is running on. The x86 based developer PC and the SDM are physically connected to the same Ethernet network, and the TCP/IP communication between the two systems is possible either by using a terminal emulator of through SFTP protocol.

After installing the virtual machine and importing the tailor-made Debian image the development system is ready to use, there is only one single setting, that the user needs to make: to set up the local IP address of the SDM provided by the DHCP server of the LAN. This IP must be set up and verified on the



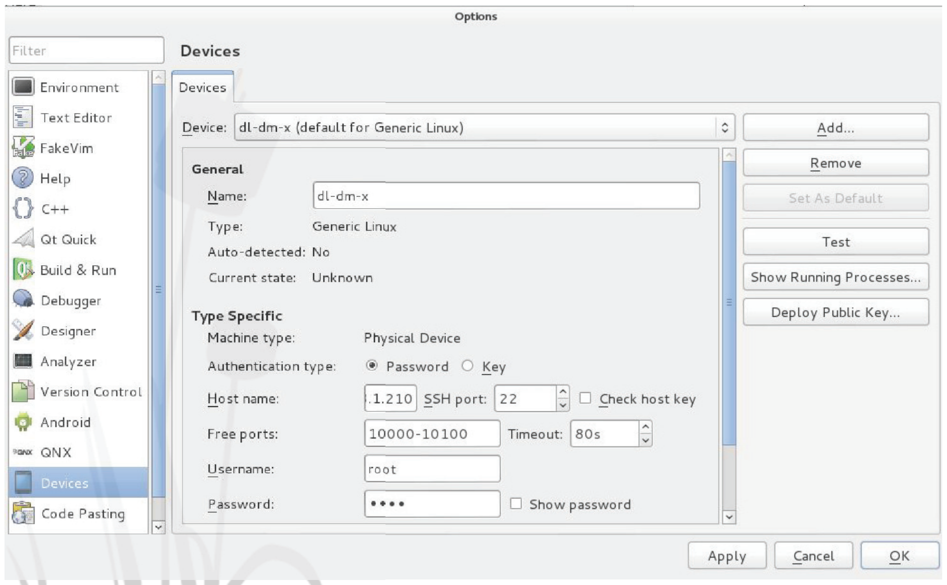
Windows alatt fut az Oracle VirtualBOX,
mely alatt a Debian Linux környezetben
a Qt Creator IDE használható az SDM programozására.

2| Cross platform development on x86 PC

development PC to be able to send the deployed ARM executable over the ethernet network by using SFTP protocol. The IP address must be written into the HOST NAME field of Qt Creator's Tools/Option/Devices menu (see figure 3). To obtain and check this IP, we need to start a small preinstalled network configuration program on the display module and set up either dynamic or static IP address for the module on the network. When dynamic IP address is used, we must be sure, that the DHCP server is enabled on the network's router. Because DHCP may provide different IP addresses to the SDM later on, it is a must to check and adjust this setting also on the development PC. To avoid such an inconvenience, it is highly recommended to use static address, and choose one high number such as 192.168.1.210 to avoid IP conflicts with DHCP addressed other TCP/IP devices on the network. Advantage of using static IP address is that no need to adjust it more than once in development PC's QT Creator. When the verification of the connection between the developer PC and the smart display module is successful, we are ready to design the HMI.

Starting Qt Creator project

There are a few types of projects can be started with Qt Creator cross platform



Devices

Device: dl-dm-x (default for Generic Linux)

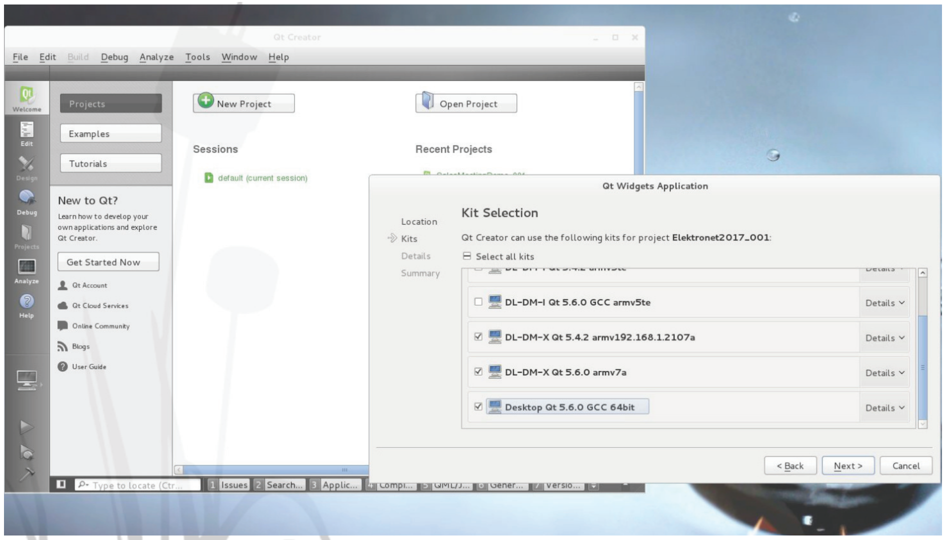
General
Name: dl-dm-x
Type: Generic Linux
Auto-detected: No
Current state: Unknown

Type Specific
Machine type: Physical Device
Authentication type: [✓ Password] Key
Host name: .1.210 SSH port: 22 Check host key
Free ports: 10000-10100 Timeout: 80s
Username: root
Password: •••• Show password

3| Making connection between developer PC and SDM development system. Some of them require graphics processor to use above Qt version 5.6 and no more software rendering is available when no GPU is present in the system. Therefore, it may cause problems when using the cheaper line of DLogic SDM module „i” family (Freescale Cortex Arm 9, iMX257), on these DLOGIC SDMs, we can only use widget-based applications. We will use this type of project in our examples. Earlier versions of the board support package had also earlier versions of Qt, so with these on board we may also create simpler Quick and Canvas 3D projects by using the provided software graphics rendering on “i” versions, if the resource requirements are not so high. The software in Qt are written in C or C++, but other language versions are also available, even there is an own language called Qt Quick.

The project type chosen by us offers a variety of graphics controls (Widget = Windows Gadget), like push buttons, vertical and horizontal sliders, checkboxes, radio buttons to design a state-of-the art HMI. These widgets can be freely placed on the canvas and their properties can be set in the graphic IDE. The required interactions, such as behavior when clicking, double clicking or sliding could be programmed as the event handlers of these controls in C++. We will show examples of these procedures later in this article.

When creating a new project, we need to select from the preinstalled kit selection



Qt Widgets Application

Kit Selection

Qt Creator can use the following kits for project **Elektronet2017_001**:

Select all kits

- [ ] DL-DM-I Qt 5.6.0 GCC armv5te
- [✓] DL-DM-X Qt 5.4.2 armv192.168.1.2107a
- [✓] DL-DM-X Qt 5.6.0 armv7a
- [✓] Desktop Qt 5.6.0 GCC 64bit

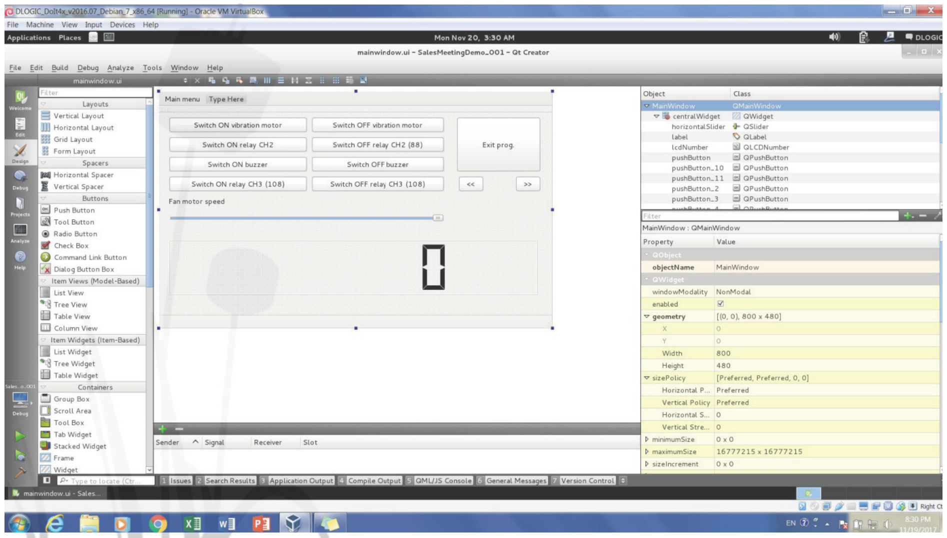
4| Selecting target hardware platforms

the target hardware we are developing to. There is a possibility to make a multiply selection, but within one project only those are included, that we defined when starting. It is recommended to always chose the local PC as a target as well as our SDM.

When starting the software project, we may design the screen by choosing „MainWindow.ui”, this is the canvas of the SDM, where we can place the widgets to be used for the human machine interface. In our example we used some push buttons, horizontal sliders, and seven segment displays. The push buttons are made to switch devices on and off through controlling some of the general purpose I/O (GPIO) ports, the sliders are used for changing the duty cycle of different pulse width modulation (PWM) controls, that can be used for speed regulation of fans or adjusting backlight brightness of the display. The segment display is to show the status of the last command.

In our example we kept the simplicity and the understanding in focus, in real applications of course more complex, professional and trendy screen designs are possible.

Before starting to explain the event handler software procedures, this is important to make an overview about accessing the hardware interfaces such as GPIOs, PWM outputs in Linux for better understanding.


mainwindow.ui – SalesMeetingDemo_001 – Qt Creator
Main menu
Type Here
Switch ON vibration motor
Switch OFF vibration motor
Switch ON relay CH2
Switch OFF relay CH2 (88)
Switch ON buzzer
Switch OFF buzzer
Switch ON relay CH3 (108)
Switch OFF relay CH3 (108)
Exit prog.
<<
>>
Fan motor speed

Object	Class
MainWindow	QMainWindow
centralWidget	QWidget
horizontalSlider	QSlider
label	QLabel
lcdNumber	QLCDNumber
pushButton	QPushButton
pushButton_10	QPushButton
pushButton_11	QPushButton
pushButton_2	QPushButton
pushButton_3	QPushButton

MainWindow : QMainWindow

Property	Value
QObject	
objectName	MainWindow
QWidget	
windowModality	NonModal
enabled	✓
geometry	[(0, 0), 800 x 480]
X	0
Y	0
Width	800
Height	480
sizePolicy	[Preferred, Preferred, 0, 0]
Horizontal P...	Preferred
Vertical Policy	Preferred
Horizontal S...	0
Vertical Stre...	0
minimumSize	0 x 0
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0

5| Graphics design of the screen of the HMI on the SDM

Using GPIO ports on DLOGIC SDMs

The Linux distribution of the DLOGIC SDMs makes it possible to control GPIOs through manipulating contents of files in a special file system (/sys/). For all available ports there is a directory containing several files describing different properties of the concerning physical device. (example: /sys/class/gpio/gpio88 for GPIO88 I/O port).

```
root@dlogic-dm:/sys/class/gpio# cd gpio88
root@dlogic-dm:/sys/class/gpio/gpio88# ls
active_low  direction  power      uevent
device      edge       subsystem  value
```

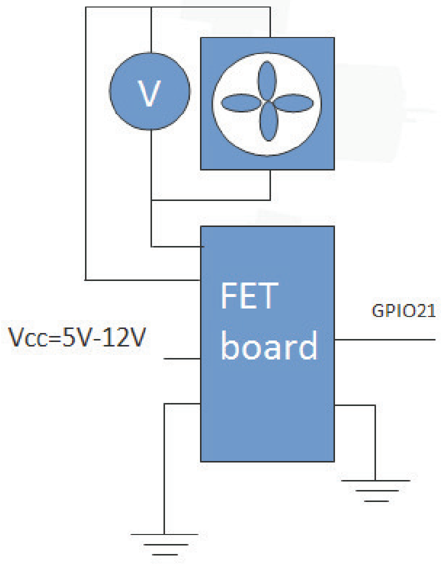
The files listed above contain the properties featuring the device and can be read or written to access or change data on the port.

To change the direction to “input” of the port, one can write the value „in” to the direction file by using command echo in > direction, or vice versa echo out > direction will make the port to be output.

The value read from or written to the port can be accessed by reading or writing the value file. In case using the echo 1> value command, we assigned logical “high” value to the output, by writing logic „0” level with command echo 0 > value we switch off the port. If we connect a power conversion device such as a FET or Relay module to the output, we can switch high power electronics devices by applying commands to the SDM. (It is important to mention that due to the 3.3V power supply of the ARM controller, the output level of the GPIOs are also limited, so voltage level corrections may be necessary by applying special components between relays and GPIOs.

Using PWM devices on DLOGIC SDMs

The pulse width modulated (PWM) outputs can be used to decrease the effective voltage of the pin by periodically switching off the power. In figure 6 a FET driver from one of the PWM output ports is responsible to cut off time to time the DC 5V..12V power



6| Controlling voltage by PMW

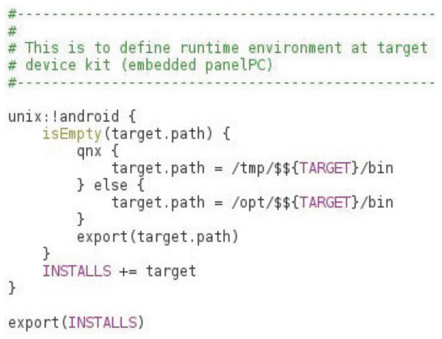
on the connectors of the fan. The average effective value of the voltage will depend on the ratio between the times of having 5V..12V and 0V on the fan connectors. Lowering the effective value of the supply voltage makes the fan speed decrease. The figure shows that the GPIO21 PWM output is responsible for controlling the switching frequency of the FET, while the power is switched by the power field effect transistor. Setting up 50% duty cycle (meaning the half of the time the full power is on the fan, and the other half of the time there is 0V) the rotation speed decreases to half. Duty cycle can be set up by writing “on” time in nanoseconds to the concerning file of the related PWM device (mxc_pwm.0):

```
root@dlogic-dm:/sys/devices/platform/mxc_pwm.0# cat dutyns
0
root@dlogic-dm:/sys/devices/platform/mxc_pwm.0# echo 100000 > dutyns
root@dlogic-dm:/sys/devices/platform/mxc_pwm.0# cat dutyns
100000
```

A special PWM device the has been assigned to the backlight module of the DLOGIC SDM, changing the duty cycle of this device will make the backlight dimmed:

```
root@dlogic-dm:/sys/class/backlight/pwm-backlight.1# cat max_brightness
500
root@dlogic-dm:~# cd /sys/class/backlight/pwm-backlight.1
root@dlogic-dm:/sys/class/backlight/pwm-backlight.1# echo 100 > brightness
root@dlogic-dm:/sys/class/backlight/pwm-backlight.1# cat brightness
100
```

The custom-made HMI examples



```
#-------------------------------------------------
#
# This is to define runtime environment at target
# device kit (embedded panelPC)
#-------------------------------------------------

unix:!android {
    isEmpty(target.path) {
        qnx {
            target.path = /tmp/$${TARGET}/bin
        } else {
            target.path = /opt/$${TARGET}/bin
        }
        export(target.path)
    }
    INSTALLS += target
}

export(INSTALLS)
```

The simple HMI solution introduced in our example is capable to control a few devices through using a few GPIO ports, as well as able to control the speed of a fan and backlight of the TFT via horizontal sliders and PWM controls. After starting the project, we need to set up the runtime environment in the „.pro” file by adding the code provided, otherwise when deploying to SDM without setting up runtime variables, we may be driven to error messages.

To control the GPIO ports we should

```
void MainWindow::on_pushButton_clicked()
{
    QString filename = "/sys/class/gpio/gpio88/value";
    QFile file(filename);
    if (file.open(QIODevice::ReadWrite )) {
        QTextStream out(&file);
        out << "1";
        ui->lcdNumber->display("088    ");
    }
}

void MainWindow::on_pushButton_2_clicked()
{
    QString filename = "/sys/class/gpio/gpio88/value";
    QFile file(filename);
    if (file.open(QIODevice::ReadWrite )) {
        QTextStream out(&file);
        out << "0";
        ui->lcdNumber->display("088 OFF");
    }
}
```

write event handlers for the onclick event of the push buttons. The switching of the port is done through writing special files in the /sys/ file system. Changing to logical level ‘1’ GPIO 88 port will switch on, while applying ‘0’ value it will switch off. The event handlers written for the “onclick” event of the concerning two buttons will do the job. The lcdNumber widget is a 7-segment display, we use it to show some status information about the last given command.

To control the speed of the fan we should

```
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    QString filename = "/sys/devices/platform/mxc_pwm0/periodns";
    QFile file(filename);
    if (file.open(QIODevice::ReadWrite )) {
        QTextStream out(&file);
        out << "40000";

    }
ui->lcdNumber->display("SET");

}
```

use a horizontal slider widget. The set up PWM frequency is 25 kHz, therefore the concerning period time should be 40 000 nsec. This value should be written into the right periodns file when initializing the program. For this we used the event generated when the MainWindow object is created.

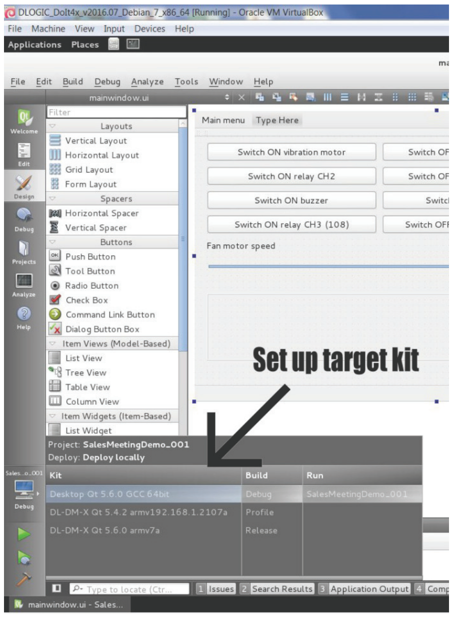
The value of the duty cycle is controlled

```
void MainWindow::on_horizontalSlider_sliderMoved(int position)
{
}

void MainWindow::on_horizontalSlider_valueChanged(int value)
{

    QString filename = "/sys/devices/platform/mxc_pwm0/dutyns";
    QFile file(filename);
    if (file.open(QIODevice::ReadWrite )) {
        QTextStream out(&file);
        out << value*800;
    }
    ui->lcdNumber->display(round(value*400));
}
```

by the horizontal sliders, which can have a value between 0-100. For the value “100” 40000 ns (100%) “on” time is connected so we need to linearly interpolate for lower values by using value*400 [ns] duty cycle for any cases. This must be programmed in the event of changing horizontal slider’s value.

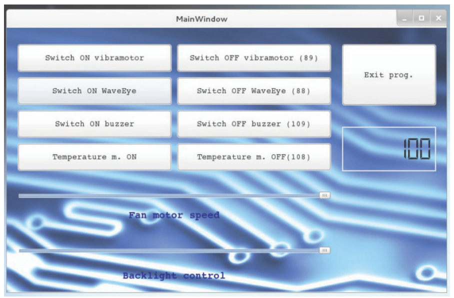


7| Select target kit for running code

Using the same methods for all controls required by the HMI we are ready to test the written program. The screenshot on figure 7 shows the possibility to define target environment for testing the code, it is recommended to select here Desktop PC at the beginning, and we can develop and debug our program on the development machine. As this PC does not have the physical peripherals of the SDM, we cannot make a full-featured test, however it is possible to check whether the concerning files are being written right.

When debugging shows no more mistakes, we may deploy the finished program to the SDM, by simple choosing the DL-DM-X kit from the selection and Qt Creator will generate now ARM executable, which will be copied to the SDM by Qt using SFTP protocol over the ethernet network.

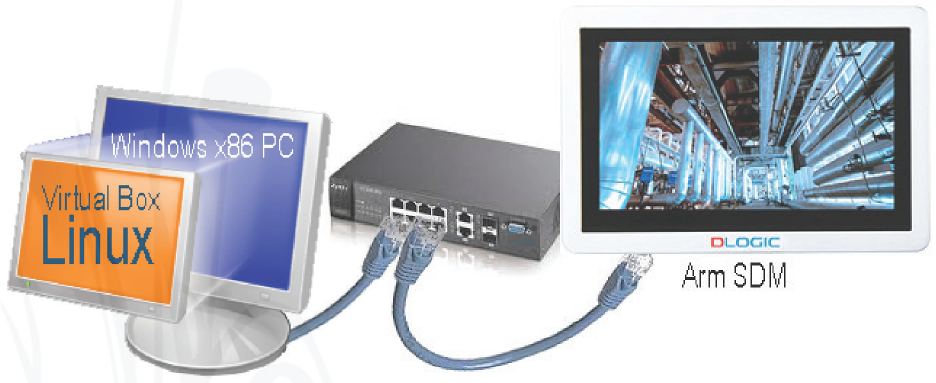
The simple HMI looks like this:



8| Simple HMI design

The advantage of the cross-platform development is that all development and debugging can be made on a desktop PC, and when finished, same tool can be used to generate and deploy executable for the ARM platform.

Applying the way of developing HMI by software adaption, a remarkable resource reduction could be accessed as well as time to market of the finish product shortens.



	TOUCH DISPLAY COMPUTERS / PANEL PCS								
	SERIES	SIZE	ORDER CODE	DISPLAY RESOLUTION	BACKLIGHT LUMINANCE [CD/M2]	CAPACITIVE TOUCH PANEL	OPTICAL BONDING	X SERIES	I SERIES
INDUSTRIAL SERIES	DL-DM430x	4.3"	101872	480 X 272	550	*		*	
	DL-DM430i	4.3"	101893	480 X 272	550	*			*
	DL-DM500x	5.0"	101873	800 X 480	550	*		*	
	DL-DM500i	5.0"	101894	800 X 480	550	*			*
	DL-DM700x	7.0"	101853	800 X 480	550	*		*	
	DL-DM700i	7.0"	101895	800 X 480	550	*			*
	DL-DM900x	9.0"	101874	800 X 480	550	*		*	
	DL-DM900i	9.0"	101896	800 X 480	550	*			*
	DL-DM1010x	10.1"	101875	1280 x 800	550	*	*	*	
	DL-DM1210x	12.1"	101876	1024 x 768	550	*	*	*	
	DL-DM1560x	15.0"	101877	1920 x 1080	550	*		*	
HYGIENE CRITICAL SERIES (STAINLESS STEEL FRONT HOUSING 316L)	DL-DM700x-H	7.0"	101878	800 x 480	550	*		*	
	DL-DM900x-H	9.0"	101879	800 x 480	550	*		*	
	DL-DM1010x-H	10.1"	101880	1280 x 800	550	*	*	*	
	DL-DM1210x-H	12.1"	101881	1024 x 768	550	*	*	*	
TOUGH GLASS SERIES – FRONT MOUNT, 4MM SAFETY GLASS, UL751 S46B	DL-DM430x-SGF	4.3"	101882	480 X 272	550	*	2 LAYER	*	
	DL-DM430i-SGF	4.3"	101897	480 X 272	550	*	2 LAYER		*
	DL-DM500x-SGF	5.0"	101883	800 X 480	550	*	2 LAYER	*	
	DL-DM500i-SGF	5.0"	101898	800 X 480	550	*	2 LAYER		*
	DL-DM700x-SGF	7.0"	101884	800 X 480	550	*	2 LAYER	*	
	DL-DM700i-SGF	7.0"	101899	800 X 480	550	*	2 LAYER		*
	DL-DM700x-SGF IO	7.0"	101885	800 X 480	550	*	2 LAYER	*	
	DL-DM900x-SGF	9.0"	101886	800 X 480	550	*	2 LAYER	*	
	DL-DM900i-SGF	9.0"	101900	800 X 480	550	*	2 LAYER		*
	DL-DM1010x-SGF	10.1"	101887	1280 x 800	550	*	3 LAYER	*	
	DL-DM1210x-SGF	12.1"	101888	1024 X 768	550	*	3 LAYER	*	
TOUGH GLASS SERIES – REAR MOUNT, 4MM SAFETY GLASS, UL751 S46B	DL-DM500x-SGR	5.0"	101889	800 x 480	550	*	2 LAYER	*	
	DL-DM500i-SGR	5.0"	101901	800 x 480	550	*	2 LAYER		*
	DL-DM700x-SGR	7.0"	101890	800 X 480	550	*	2 LAYER	*	
	DL-DM700i-SGR	7.0"	101902	800 X 480	550	*	2 LAYER		*
	DL-DM900x-SGR	9.0"	101891	800 X 480	550	*	2 LAYER	*	
	DL-DM900i-SGR	9.0"	101903	800 X 480	550	*	2 LAYER		*
	DL-DM1010x-SGR	10.1"	101892	1280 x 800	550	*	3 LAYER	*	

	TOUCH DISPLAY MONITORS								
	SERIES	SIZE	ORDER CODE	DISPLAY RESOLUTION	BACKLIGHT LUMINANCE [CD/M2]	CAPACITIVE TOUCH PANEL	OPTICAL BONDING	LVDS	HDMI
INDUSTRIAL SERIES	DL-DM700-LVDS	7.0"	101676	800 x 480	550	*	1 LAYER	*	
	DL-DM900-LVDS	9.0"	101711	800 x 480	550	*	1 LAYER	*	
	DL-DM1010-LVDS	10.1"	101705	1280 x 800	550	*	2 LAYER	*	
	DL-DM1210-LVDS	12.1"	101737	1024 x 768	550	*	2 LAYER	*	
	DL-DM1010-HDMI	10.1"	101803	1280 x 800	550	*	2 LAYER		*
	DL-DM1210-HDMI	12.1"	101694	1024 x 768	550	*	2 LAYER		*
TOUGH GLASS SERIES – FRONT MOUNT, 4MM SAFETY GLASS, UL751 S46B	DL-DM700-LVDS-SGF	7.0"	101794	800 x 480	550	*	2 LAYER	*	
	DL-DM900-LVDS-SGF	9.0"	101795	800 x 480	550	*	2 LAYER	*	
	DL-DM1010-LVDS-SGF	10.1"	101796	1280 x 800	550	*	3 LAYER	*	
	DL-DM1210-LVDS-SGF	12.1"	101797	1024 x 768	550	*	3 LAYER	*	
	DL-DM1010-HDMI-SGF	10.1"	101804	1280 x 800	550	*	3 LAYER		*
	DL-DM1210-HDMI-SGF	12.1"	101805	1024 x 768	550	*	3 LAYER		*
TOUGH GLASS SERIES – REAR MOUNT, 4MM SAFETY GLASS, UL751 S46B	DL-DM700-LVDS-SGR	7.0"	101799	800 x 480	550	*	2 LAYER	*	
	DL-DM900-LVDS-SGR	9.0"	101800	800 x 480	550	*	2 LAYER	*	
	DL-DM1010-LVDS-SGR	10.1"	101801	1280 x 800	550	*	3 LAYER	*	
	DL-DM1210-LVDS-SGR	12.1"	101802	1024 x 768	550	*	3 LAYER	*	
	DL-DM1010-HDMI-SGR	10.1"	101806	1280 x 800	550	*	3 LAYER		*
	DL-DM1210-HDMI-SGR	12.1"	101807	1024 x 768	550	*	3 LAYER		*

endrich