# Практика применения в устройствах на микроконтроллерах дисплейных модулей от 4D Systems

## Часть 2

**Павел Редькин (г. Ульяновск)**

**В первой части статьи рассказывалось об основных характеристиках, базовых функциях и системе команд режима Serial дисплейных модулей от компании 4D Systems на примере одного из таких модулей – µOLED-128-G2. Там же был описан макет устройства на базе микроконтроллера (МК) и дисплейного модуля с реализованным самодостаточным пользовательским интерфейсом.**
**Во второй части статьи приводится информация, позволяющая усовершенствовать указанный интерфейс, повысить его привлекательность и расширить возможности в плане текстового вывода за счёт использования медийных функций дисплейного модуля. Для этих целей необходимо применение дополнительных инструментальных средств.**

### Инструментальные средства разработки-отладки

Если в приложении требуется поддержка каких-либо медийных функций дисплейного модуля, то для их начальной настройки необходимо задействовать специальные программные инструментальные средства. Все они являются бесплатными и свободно распространяются производителем. Основное из них – интегрированная система разработки-отладки Workshop4 IDE, которую можно загрузить по ссылке с интернет-страницы [1].

Workshop4 IDE представляет собой программный пакет, работающий под ОС Microsoft Windows и интегрирующий редактор исходных текстов, компилятор, компоновщик и загрузчик для разработки и записи в память графических процессоров управляющих программ на языке 4DGL. Помимо этого, Workshop4 IDE поддерживает запись на µSD-карты памяти шрифтов и графических объектов пользователя, а также обновление «прошивок» графических модулей и эмуляцию управляющего МК. Функции Workshop4 IDE, связанные с записью в память графических процессоров модулей, доступны только при наличии специального аппаратного инструментального обеспечения – программного адаптера µUSB-PA5 от 4D Systems. Он представляет собой преобразователь уровней USB–UART и подключается к модулю µOLED-128-G2 через один ряд в двухрядном разъёме последнего. Помимо управляющих сигналов µUSB–PA5 обеспечивает также питание модуля. Внешний вид адаптера µUSB–PA5 представлен на рисунке 6.

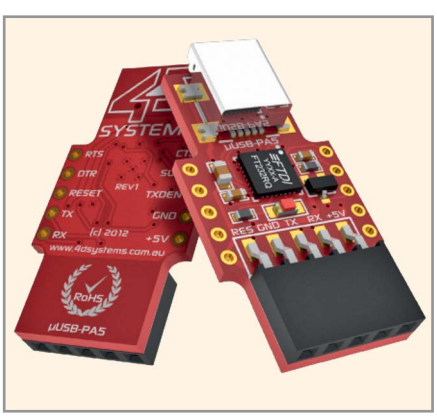


**Рис. 6. Внешний вид программного адаптера µUSB-PA5**

Исходный вид главного окна программной среды Workshop4 IDE приведён на рисунке 7.

### Задание шрифтов для вывода текста

Для обеспечения возможности задания в приложении дополнительных шрифтов необходимо в Workshop4 IDE создать ViSi-проект. В общем случае в режиме ViSi осуществляется написание, отладка, компоновка, компиляция и загрузка в память графических процессоров программ на языке 4DGL. Здесь мы используем режим ViSi для генерации нескольких констант, подлежащих затем переносу в управляющую программу нашего МК, а также для записи пользовательских данных на карту памяти.

Для создания ViSi-проекта в главном окне Workshop4 IDE следует выбрать Create a New Project, после чего задать целевой дисплейный модуль, в нашем случае – µOLED-128-G2, как показано на рисунке 8. Физическое подключение самого модуля к компьютеру нам не требуется. Выбрав модуль, кликаем мышью на кнопке Next со стрелкой, далее в открывшемся окне выбора вида проекта – на кнопке со стрелкой около иконки ViSi, изображённой на рисунке 9. Открывается окно редактора, в центре которого располагается созданный по умолчанию шаблон исходного кода программы на 4DGL, справа вверху – поле дисплея выбранного модуля, справа внизу – область диспетчера объектов проекта (см. рис. 10). Созданный ViSi-проект можно сразу сохранить, например, под именем Font_Demo. Далее необходимо в шаблоне исходного кода установить курсор в позицию, отмеченную на рисунке красной стрелкой. Так мы задаём место в программе, куда затем будут вставлены автоматически сгенерированные программные фрагменты. Заметим, что целиком эта программа нам сейчас не понадобится, мы будем использовать только несколько констант из неё. Для генерации требуемых фрагментов выбираем в главном меню опцию Widgets, а в ней – вкладку Labels. С вкладки перетаскиваем мышью на область дисплея нашего модуля иконку String, после чего выбираем в перечне параметров диспетчера объектов для объекта Strings1 позицию Strings, как показано на рисунке 11. В столбце Value этой позиции кликаем на кнопке выпадающего меню (кнопка с многоточием), после чего открывается окно редактора шрифтов Strings Editor, представленное на рисунке 12, в поле которого Font выбираем из выпадающего меню нужный нам шрифт. Для выбора доступны


**Рис. 7. Исходный вид главного окна среды Workshop4 IDE**

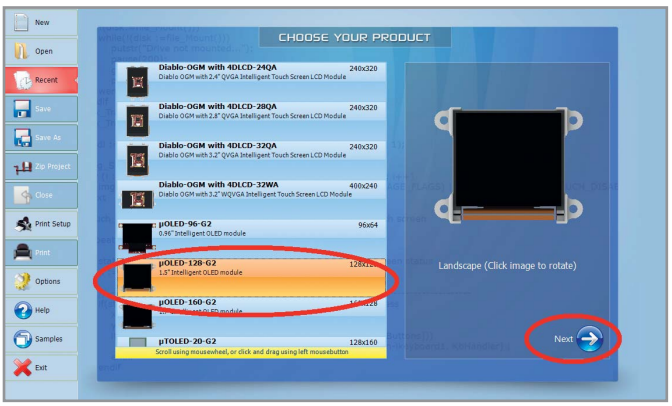


**Рис. 8. Задание целевого дисплейного модуля при создании проекта**


**Рис. 9. Выбор вида проекта – ViSi-проект**

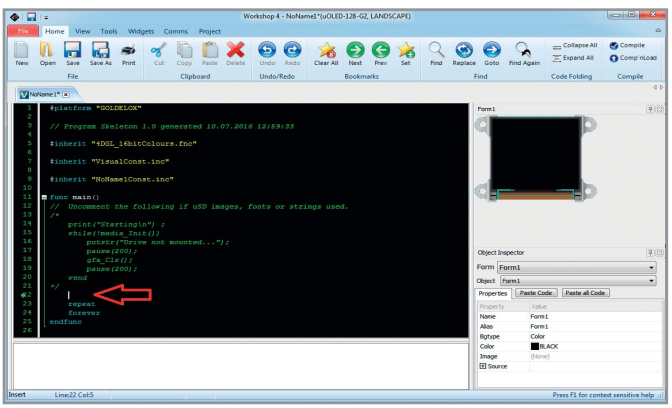

**Рис. 10. Окно редактора с шаблоном исходного кода ViSi-проекта**

все шрифты Windows. В поле Size задаём нужный размер шрифта. Необходимо учитывать, что выбранный шрифт в дальнейшем будет сохранён в памяти µSD-карты именно с таким размером. Кликнув на кнопке OK, возвращаемся в окно редактора. В заключение кликаем на кнопке Paste Code диспетчера объектов, после чего в ранее заданной курсором области редактора появляется фрагмент кода, сгенерированный в результате всех действий по выбору шрифта. Указанные действия можно повторить несколько раз, выбрав, таким образом, несколько шрифтов и сгенерировав несколько последовательных фрагментов кода (см. рис. 13).

Затем необходимо физически подключить к компьютеру µSD-карту, воспользовавшись, например, картридером. После того как её обнаружит ОС Windows, нужно обычным образом отформатировать её средствами Windows в файловой системе FAT с заданными по умолчанию параметрами. После подготовки карты выбираем в главном меню опцию Home, в ней запускаем команду компиляции исходного текста Compile и ожидаем завершения компиляции проекта. По окончании компиляции внизу, в окне сообщений, появляются её результаты. При отсутствии ошибок (сообщение в окне – No errors) открывается окно выбора носителя для копирования данных, изображённое на рисунке 14. Задаём в поле окна Drive нашу карту (в компьютере автора это диск O), кликом на кнопке OK запускаем запись, ход которой отобра-

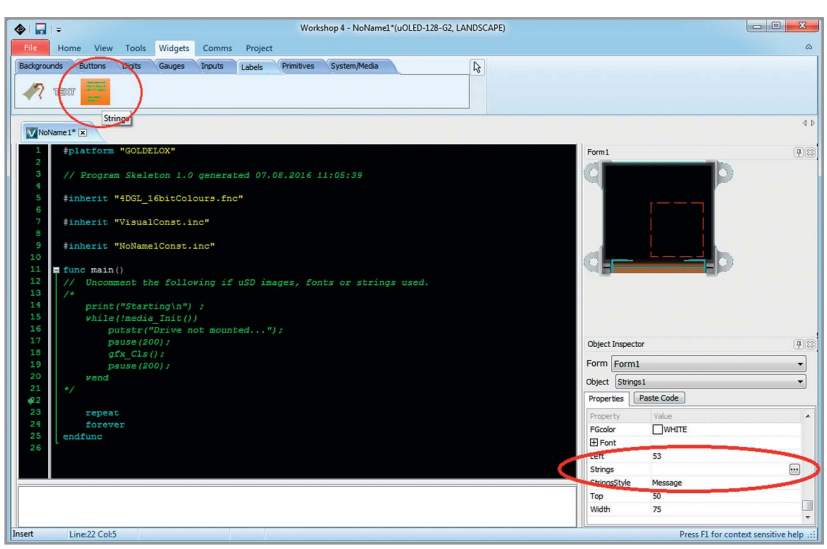

**Рис. 11. Создание объекта «Текстовый шрифт» Strings1**

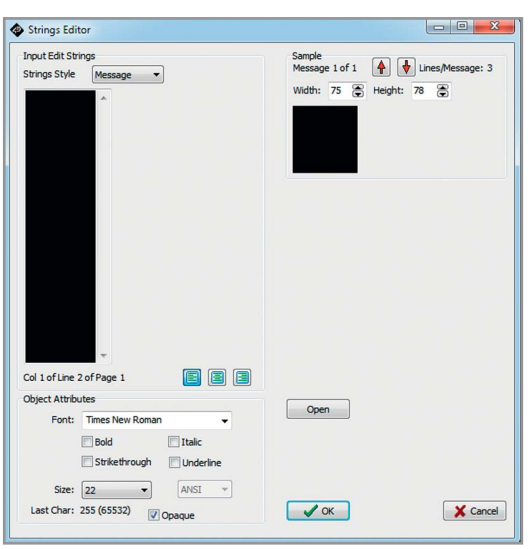


**Рис. 12. Окно редактора шрифтов Strings Editor**

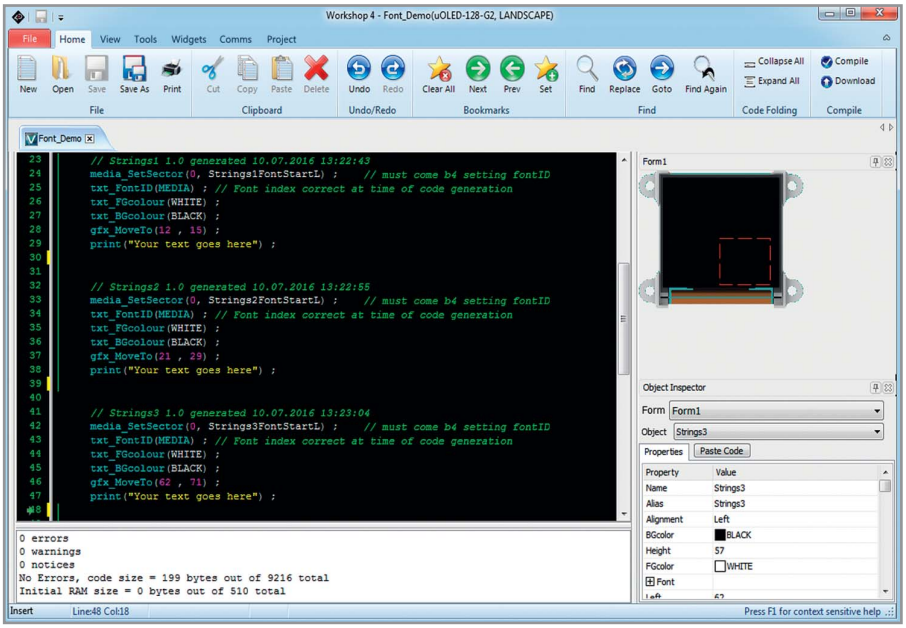


**Рис. 13. Фрагменты исходного кода, сгенерированные в результате выбора нескольких шрифтов**

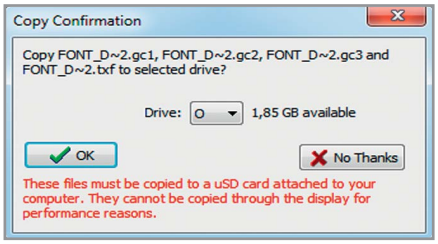


**Рис. 14. Окно выбора носителя для копирования данных мультимедиа**

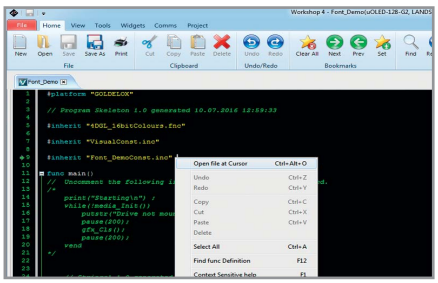

**Рис. 15. Открытие заголовочного файла констант через контекстное меню**

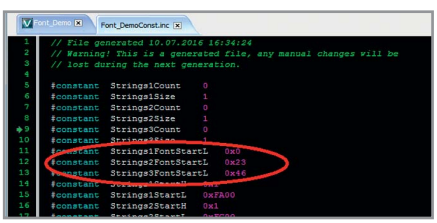

**Рис. 16. Заголовочный файл констант ViSi-проекта**

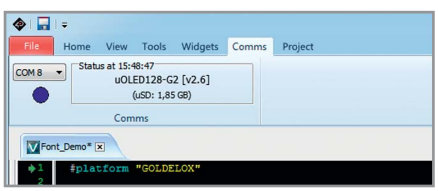

**Рис. 17. Установленное соединение между компьютером и графическим модулем при эмуляции МК**

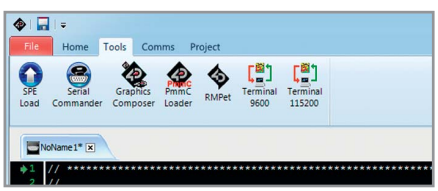

**Рис. 18. Программные инструментальные средства для обмена данными между компьютером и графическим модулем**

жается в окне индикатора прогресса. Так как при форматировании карты нами была задана файловая система FAT, перед началом записи Windows спросит разрешение изменить файловую систему носителя с FAT на RAW, поскольку объекты мультимедиа записываются на носитель в формате RAW. Разрешение нужно дать. После завершения записи необходимо получить данные о местонахождении записанных объектов (шрифтов) в адресном пространстве носителя. Для этого в начале исходного кода нашего проекта находим ссылку на автоматически сформированный заголовочный файл констант #inherit «Font_DemoConst.inc», устанавливаем на ней курсор и кликаем правой кнопкой мыши. В открывшемся контекстном меню выбираем опцию Open File at Cursor, как показано на рисунке 15, после чего указанный файл открывается во вкладке окна редактора, представленного на рисунке 16. Значение константы Strings1FontStartL из этого файла представляет собой адрес сектора на карте памяти, где хранится первый из заданных нами шрифтов, Strings2FontStartL – адрес сектора хранения второго шрифта и т.д. Указанные значения выделены на рисунке красным овалом. Используем их в качестве параметров команды media_SetSector в режиме Serial. Необходимо только учитывать, что параметры в этой команде двухбайтовые, поэтому значения констант должны быть дополнены старшими незначащими нулями до размера слова.

После получения нужных значений адресов созданный в ViSi проект сохраним и закроем. В дальнейшем он понадобится нам для работы с изображениями и видео. Для получения дополнительной информации по заданию шрифтов рекомендуется обратиться к [2].

Использовать полученные в ViSi-проекте данные можно следующим образом. Во-первых, сразу вставить значения констант адресов непосредственно в параметры библиотечной функции media_SetSector в исходном коде управляющей программы МК с тем, чтобы в дальнейшем использовать эту функцию для задания шрифтов в приложении. Никакие дополнительные аппаратные средства (кроме, разумеется, средств программирования-отладки МК) для этого не нужны. Однако если в нашем распоряжении имеется адаптер µUSB-PA5, можно средствами Workshop4 IDE осуществить предварительную эмуляцию управляющего МК на компьютере, что значительно облегчит отладку кода в МК.

## Эмуляция управляющего МК в Workshop4

Чтобы перейти к эмуляции МК, необходимо физически подключить дисплейный модуль к порту USB компьютера через µUSB-PA5. Для поддержки эмуляции медийных команд в модуль должна быть предварительно вставлена µSD-карта с записанными на неё медийными данными, например, шрифтами. Для эмуляции всех остальных команд необходимости в µSD-карте нет. После обнаружения ОС Windows USB-подключения адаптера следует уточнить номер созданного в системе виртуального COM-порта. Далее необходимо в среде Workshop4 IDE создать Serial-проект, для чего в окне выбора вида проекта (см. рис. 9) нужно кликнуть на кнопке со стрелкой в области Serial, после чего откроется окно редактора, содержащее на

вкладке пустой шаблон. В опции главного меню Comms созданного проекта задаём номер нашего COM-порта из выпадающего списка, как это показано на рисунке 17. После установки соединения между компьютером и модулем кружок в поле Comms сменит цвет с красного сначала на жёлтый, а потом на синий, и рядом с ним появится информация о подключённом модуле и о вставленной в него карте памяти (при наличии последней). Далее в главном меню Tools выбираем эмулятор – инструментальное средство Serial Commander (см. рис. 18), на вкладках которого сгруппированы все доступные команды режима Serial, как показано на рисунке 19. Найдя на вкладках эмулятора нужную команду, отмечаем её мышью и передаём в модуль кликом на кнопке Send. Если для команды предусмотрены параметры, то следует предварительно задать их в соответствующих полях внизу вкладки в десятичном виде. В поле справа от вкладок отображаются имена, cmd-коды, значения времени обработки в секундах, подтверждения и ответы для всех команд, переданных в текущем сеансе эмуляции.

Для вывода на дисплей текста с заданным шрифтом действуем в следующем порядке. Из вкладки Gfx передаём команду очистки дисплея gfx_Cls. Чтобы отменить заданную по умолчанию прокрутку дисплея, из вкладки Other передаём команду SSTimeout с параметром 0 ms. Затем из вкладки Media передаём команду начальной инициализации карты памяти media_Init. Если логикой модуля карта обнаруживается и успешно инициализируется, то ответ модуля на эту команду должен быть ненулевым. Нулевой ответ означает, что карту инициализировать не удалось и выполнение последующих связанных с ней команд невозможно. Убедившись в корректной инициализации карты, из той же вкладки передаём команду media_SetSector, задав для неё значения старшего (ноль) и младшего (например,

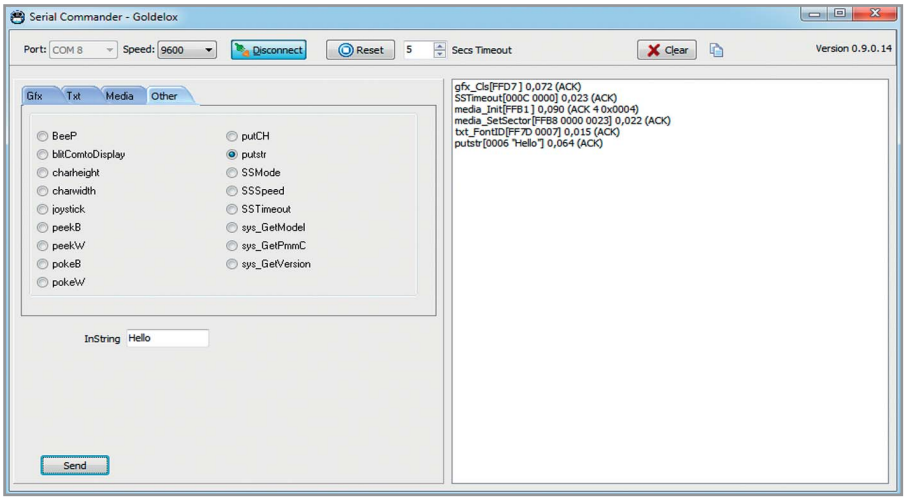


**Рис. 19. Эмулятор Serial Commander, содержащий команды режима Serial**

константа Strings2FontStartL в десятичном виде) слов адреса сектора. Затем из вкладки Txt передаём команду txt_FontID, задав для неё в поле FontNumber значение 7. FontNumber = 7 означает, что шрифт загружается с носителя, то есть с карты памяти. На этом, собственно, задание шрифта завершается. Теперь переходим непосредственно к выводу текстовых данных в выбранном шрифте на дисплей. Из вкладки Other передаём команду вывода строки текста txt_putstr, предварительно задав в поле InString какую-нибудь текстовую строку, например, Hello. Заданная строка в выбранном шрифте должна появиться на дисплее модуля. Вся перечисленная последовательность команд отображена в поле справа от вкладок на рисунке 19.

Заметим, что в случае использования загружаемых с носителя шрифтов придётся экспериментально подбирать нужный размер текстовых символов с помощью соответствующих команд режима Serial и с учётом сохранённого на носителе размера шрифта.

Для задания цвета выводимого текста удобно использовать данные из файла цветовых 16-разрядных констант, ссылка на который #inherit «4DGL_16bitColours.fnc» по умолчанию имеется в заголовочной части шаблона исходного текста проекта (см. рис. 15). Этой файл может быть открыт для просмотра с помощью опции Open File at Cursor из его контекстного меню.

Заметим, что в главном меню Tools ViSi-проекта имеются также инструменты SPE Load и PmmC Loader (см. рис. 18), позволяющие обновить текущие версии программы SPE и «прошивки» PmmC модуля на актуальные версии, содержащиеся в Workshop4 IDE. Команда SPE Load может быть также использована для восстановления стёртой программы SPE. Последняя стирается из памяти графического процессора всякий раз при загрузке туда пользовательской программы из ViSi-проекта.

Примерно такая же последовательность команд была сформирована автором в управляющей программе МК Uart_4D.c. Результаты работы программы можно видеть на рисунке 20. Задание нового шрифта из трёх записанных на карту памяти по кольцу производится в программе при каждом нажатии на кнопку SW1.

## Литература

1. www.4dsystems.com.au/product/4D_Workshop_4_IDE.
2. 4D Systems Application Note 4D-AN-00084. Serial Goldelox Displaying Third Party Fonts.

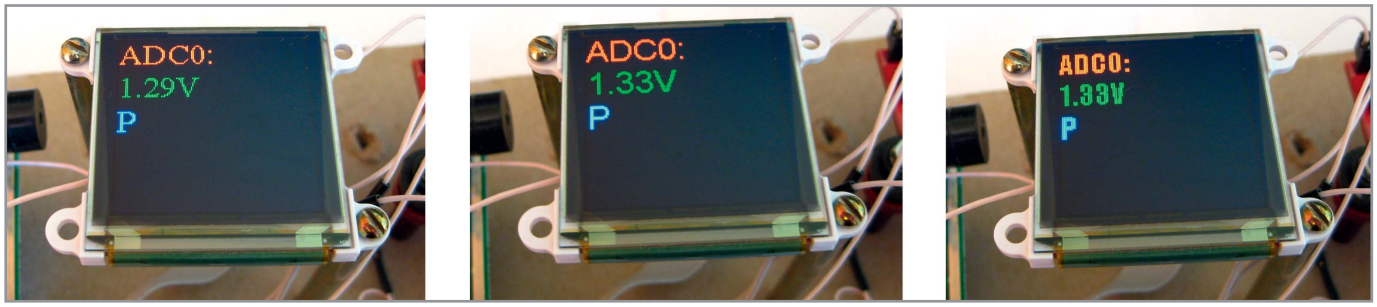


**Рис. 20. Внешний вид дисплея при выводе текстовых данных с различными записанными на карту памяти шрифтами**