

TFT | DLED | GRAPHIC | CHARACTER | UWVD | SEGMENT | CUSTOM

# Application Note FAN4224

## Working with the E35RD-MW420-C

This application note will present the firmware to drive the E35RD-MW420-C TFT Display Module with an STM32H747I-DISCO microcontroller development board.





## Contents

Introduction	6
Hardware Requirements	7
Development Tools	8
STM32CubeIDE	8
Project Development	9
Start New Project	9
Select the MCU	.10
Name the Project	.11
Add the Files to the Project	.14
Modify main.h and main.c	.17
Change Optimization Settings	.19
Build the Project	.20
Creating the Firmware Files for the E35RD-MW420-C	.21
Create the E35RD-MW420-C Header and Source Files	.21
Editing the Header File	.22
Create and Edit the ST7701S Header File	.23
Editing the Source File	.24
Setting up Arrays for the Command Paramters	.24
Initial Parameter Configuration	.25
Command or Video Mode and Additional Configuration	.25
Configuring the Initialization Sequence	.27
Additional Comments	.30
Summary	.30
LCD Handling Precautions	.31
Disclaimer	.31
Revision History	.31





## List of Figures

Figure 1: 35RD-MW420-C, KAB-M20SD-01 Adapter, and STM32H7 Disco Running Demo	5
Figure 2: E35RD-MW420-C 3.5" MIPI TFT Display	6
Figure 3: STM32H747I-Disco Development Board	7
Figure 4: Focus LCDs MIPI DSI Adapter Board v1.0	7
Figure 5: STM32CubeIDE Integrated Development Environment	8
Figure 6: Start a New Project	9
Figure 7: Target Selection Window	9
Figure 8: Select the STM32H747XIH6 Device	10
Figure 9: Selecting the STM32H747XIH6 Device in Features Section	10
Figure 10: Select "Next" to Continue Project Creation	11
Figure 11: Naming the Project	11
Figure 12: Library Package Setup	12
Figure 13: Open the Pin Configuration Perspective	12
Figure 14: Memory Protection Unit Enable	13
Figure 15: STM32CubeIDE Pin Perspective	13
Figure 16: C/C++ Perspective	14
Figure 17: Project Explorer before Refresh	15
Figure 18: Right-click Menu: Refresh	15
Figure 19: Refresh of Inc Folder	15
Figure 20: Refresh the Src Folder	16
Figure 21: Main.h Modifications	17
Figure 22: Main.c Modifications	18
Figure 23:Project Properties Main Menu Item	19
Figure 24: C/C++ Build Settings	20
Figure 25: Optimization Level	20
Figure 26: New Header File	21
Figure 27: New Source File	21
Figure 28: Include Guard	22
Figure 29: Include Headers	22
Figure 30: API	22
Figure 31: ST7701S Header File Include Guard	23
Figure 32: Page/Bank Change Command and Data	23
Figure 33: Register Defines	23
Figure 34: Bit Defines	24
Figure 35: Constant Name and Macro Alias	24
Figure 36: Arrays for Command Pages	25
Figure 37: DSI Video Mode Setting	25

# FAN4224



Figure 38: Adjusted Height and Width	26
Figure 39: E35RD-MW420-C Display Timings from the Display Initialization File	26
Figure 40: Display Timings	26
Figure 41: Switch Command Pages	27
Figure 42: Display Initialization Long and Short Writes	27
Figure 43: runInitSeqLCDConfig() Function	28
Figure 44: A Sample of the runInitSeqLCDConfig() Function	28
Figure 45: Page Switch Long Write Command	28
Figure 46: E35RD-MW420-C Running the Focus LCDs Demo	29

# FAN4224



## The E35RD-MW420-C Firmware Project

This application note will present the hardware and walk through developing the firmware required to drive the E35RD-MW420-C MIPI TFT Display with an STM32H747I-DISCO microcontroller board from ST Microelectronics. Driving a MIPI DSI display with a microcontroller presents a few challenges. The STM32H747 has the required bandwidth and I/O pins but lacks enough internal SRAM for a full frame buffer. The DISCO board presented has an external SDRAM chip used for implementing the display frame buffer.

Previous application notes (FAN4221, FAN4222, and FAN4223) had discussed the hardware required, an overview of the firmware, and how to modify the firmware from the E43RB1-FW405-C MIPI TFT Display for the E50RA-I-MW490-C MIPI TFT Display. FAN4224 develops the firmware for the E35RD-MW420-C while reusing the source files common to the MIPI displays.



Figure 1: 35RD-MW420-C, KAB-M20SD-01 Adapter, and STM32H7 Disco Running Demo





## Introduction

Developing the firmware for the E35RD-MW420-C MIPI TFT Display will be made simpler by reusing the source files for the E43GB-I-FW405-C (provided by Focus LCDs upon request) and making similar modifications as presented in FAN4223. Like the previous application notes, it is assumed that an adapter PCB will be used to connect the display to the development board. Again, the adapter PCB can be designed by the end user, or the Focus LCDs prototyping adapter board can be used for the project.

Contact Focus LCDs for more information on the Adapter Board and firmware.

The E35RD-MW420-C display used in this application is a 3.5" TFT with a 480 x 800 RGB pixel resolution from Focus LCDs. This display is interfaced over a 2-lane MIPI DSI protocol with a 20-pin FPC cable. An ST Microelectronics STM32H747XI microcontroller, specifically the Discovery Development Board, will drive the display.

The main features of the E43RB1-FW405-C are:

- 3.5" Diagonal Display, 480 x 800 RGB Pixel Resolution
- Up to 65k/262k/16.7M (24-bit) Color
- 2-Lane MIPI DSI Interface with 20-pin FPC Cable
- Transmissive/Normally Black Display Mode
- White LED Backlight
- ST7701S Display Controller
- Capacitive Touch Panel () Touch Mode: 5-Point and Gesture
- Typical Operating Voltage: 3.3V







### Hardware Requirements

The development board that will drive the display is an STM32H747I-DISCO Discovery Kit. The Kit comes with its own display but for this app note, the stock display will be replaced with the E35RD-MW420-C. See FAN4221 for more information.



Figure 3: STM32H747I-Disco Development Board

In this application, the Focus LCDs MIPI DSI Adapter Board v1.0 is used to adapt from the Q Strip connector to an FFC connector on the PCB. This adapter board is still in development. Contact Focus LCDs for more information.



Figure 4: Focus LCDs MIPI DSI Adapter Board v1.0





## Development Tools STM32CubeIDE

Developing applications typically will use STM32CubeMX for setting up peripheral initialization and STM32CubeIDE (integrated development environment) to code the application side. These tools can be downloaded from the ST website. In this application, only the STM32CubeIDE will be used in the development of the firmware.

After installation, STM32CubeIDE might request additional downloads depending on the device series integrated into the project. When developing the firmware, an additional download for the H7 series of devices was required.

arkanae, 1311 - 575025.dodD5															- C X
His hill Severa Reflector Navigate Search Propert Run Weidler	- Hulp I Hole Lan	a construction of the construction													A (10) T
DTHUSTSTICKING WTDTET		A claim to De statutores	110.0010											1007	4 8 4
(b) Protect Study of the stu												Tee!	den 2 (19. Sold S	reprint Generalise en estilea.	- 0
		Problems x 2 Jula Console -	Protection				9 I + 0		· × · Sterie St	et indicer (R.C	volomatic Come	and a			×1=0
		0 torns		142	10000				-						
		Description	Resource	Path.	Location	Abe		Menory Regions	Mersory Details				10001	100000000	
								Region	Start address	EAU address	Sex	Free	Used	Usage //O	
								-							

Figure 5: STM32CubeIDE Integrated Development Environment





## **Project Development**

In this section, the initial project generation will be presented. The steps to create a new project, what files are to be added, initial modifications, and test building the firmware will be shown.

#### **Start New Project**

Open STM32CubeIDE and select "Create a New STM32 Project" from the available selections.

	tor Navigate	Search Project	t Run \	Vindow He	lp 💄 H	ello Tony
🖻 • 🗑 🐚 🛛 • 🐔	-1001	x 1 @ 1 👩 -	63 • C	• @ •	¢ • 0	- 🤹 - 🙋
🍐 Project Explorer 🗙				E	\$ 8	8 - 0
There are no projects in yo	ur workspace.					
To add a project:						
Create a new Makefi	e project in a di	rectory containir	ng existing	ode		
Create a New STM32	project -S	elect				
	Digitation an	Existing STM320	CubeMX Co	nfiguration F	ile (.ioc)	
and the state of the second states of						
Creates a new STM3	CMake Project					
Creates a new STM3	CMake Project					

Figure 6: Start a New Project

The IDE will perform a short download, then open the product selection window.

UMPU Pitters									
* 🗃 🛱 0	710	num Mich	Dagan (	Inch & Restauces	CAD Resources		Cafashour		13 Ref
Commercial V	2								
0									
				-		6	B		
PRODUCT INFO Y				1 march			5		
Begnert >		1.			6 C	Wireles	S		
Satas >				13	8	Connec	CONICY		
100			-TH32						
			12.	ALCON S			- A -		
Marketing Status									
Pice 7			100			OTH			
Farmer and the second se						SIIM	96 J		
Package 2									
Package 2									
Package 2 Core 2									
Pachage > Care > Coprocessor > WEMORY ~ Flank Frem 6 to 4096 (40 ytm)		STM3 Low-P	2WL3 No ower &	ew Sub- Flexibili	lGHz So ity	с		57	
Datage         2           Care         >           Capocessor         >           MEMORY            Trank From 3 to 40% (dbytec)         40%           PEPDM From 3 to 10% (dbytec)         40%	MCOs/BPOs L	STM3: Low-P	2WL3 No Power &	<del>ew Sub-</del> Flexibili	l GHz So ity	с	1	57	
Package         2           Care         >           Capacessor         >           MEMORY         >           Defines         From 0 to 4006 (Mbytec)           Defines         From 0 to 1004 (Bytec)           Defines         From 0 to 1004 (Bytec)	MCGW/MPOs D	STM3: Low-P	2WL3 No Power &	<del>ew Sub-</del> Flexibili	l GHz So ity	с	[	577	₫ Expt
Dockapt         2           Core         2           Dopocessor         3           MEMORY	McostaPos L	STM3 Low-P	2WL3 No Ower &	ew Sub- Flexibili	IGHz So ity	C 22 kBytes	6 kBytes	<b>577</b>	1 Expi
PACKage         2           Cop         2           Copocessor         2           MEMODRY         4006           EXPROM         4006           EXPROM         15334           RAM Total From 2 to 15024 (Bytes)         15334	McoultiPlus L	STM3: Low-P	2WL3 No OWEI & STR02001 Come STR02001 Adve	ew Sub- Flexibili	IGHz So ity	22 kBytes 32 kBytes	6 kBytes 6 kBytes	<b>577</b>	€ Expl 40 MPsc 40 MPsc
Dackage         2           Cran         2           Copocessor         3           MEMORY         -           EPROM         From 0 to 4006 (Abytec)           4000         4000           All fold From 2 to 2005 (Abytec)         4000           All fold From 2 to 2005 (Abytec)         3000	McostaPos L	STM3: Low-P	2WL3 No OWEI & STAJOOL Comp STAJOOL Advs STAJOOL Advs	ew Sub- Flexibili Ster Na 9213 Stor NA	IGHz So ity	22 ABytes 32 ABytes 32 ABytes 16 ADytes	6 kBytes 6 kDytes 6 kDytes	<b>577</b>	€ Expl 40 MPtc 40 MPtc 40 MPtc
Dockage         2           Cop         2           Copocessor         2           MEMORY         4996           0         5000 (bb/ytec)           0         3005           0         2         3005           0         4996         4996	McOstaPOs L	STM3: Low-P	2WL3 No ower &	ew Sub- Flexibili	1 GHz So ity	C 22 40ytes 32 40ytes 33 40ytes 34 40ytes 34 40ytes	6 kBytes 6 kBytes 6 kBytes 6 kBytes 6 kBytes	577 50 50 50 51 51 52 53 53 55	€ Expt 40 MP4c 40 MP4c 40 MP4c 40 MP4c 40 MP4c
Dockapt         2           Core         >           Copocessor         >           MEMORY		STM3 Low-P st: 2020 terms maccore maccore maccore maccore maccore maccore maccore	2WL3 No OWEI & STRUZOS Com STRUZOS Active STRUZOS Active STRUZOS Active STRUZOS Active STRUZOS Active STRUZOS Active	ew Sub- Flexibili son NA 0.223 son NA 0.3356 0.3355	1 GHz So ity	C 22 45/45 22 45/45 23 45/45 15 45/45 16 45/45	6 18344 6 183444 6 183444 6 183444 6 183444 6 183444 6 183444 6 183444	50 50 10 10 10 10 10 10 10 10 10 10	€ Expt 40 MP42 40 MP42 40 MP42 40 MP42 40 MP42 40 MP42 40 MP42
Package         2           Cape         2           Capecessor         2           Capecessor         2           VEMORY         4256           Package         4256           PROM. From 5 to 4206 (Adlytes)         4256           PROM. From 5 to 1005 (Adlytes)         10334           RAM. Total From 2 to 3256 (Adlytes)         3255           RAM. From 2 to 3255 (Adlytes)         30255           CCM.DAM. From 3 to 12 (Adlytes)         30256		STM3: Low-P ar: 2000 terms ar: 2000	2WL3 No OWEI & STRUCCI. Comp STRUCCI. Active STRUCCI. Active STRUCCI. Active STRUCCI. Active STRUCCI. Active STRUCCI. Active STRUCCI. Active STRUCCI. Active STRUCCI. Active	Sub-           Flexibili           State           0.213           soon FM           0.213           soon FM           0.3116           0.3355           0.3355           soon FM	1GHz So ity W.CP 12 W.CP 12 W.CP 12 SSOF42 TSSOF42 TSSOF42 UPDP14	2 245ytes 32 45ytes 32 45ytes 34 45ytes 16 45ytes 16 45ytes 16 45ytes	6 kBytes 6 kBytes 6 kBytes 6 kBytes 6 kBytes	<b>1 10</b> 50 10 15 15 15 16 15 18	Expl 40 MP42 40 MP42 40 MP42 40 MP42 40 MP42 40 MP42 40 MP42 40 MP42 40 MP42
Discage         2           Creat         2           Coprocessor         2           MEMORY         4950           EXEPTION         From 0 to 4006 (kBytes)           0         4950           EXEPTION         From 0 to 4006 (kBytes)           0         4950           EXEPTION         From 0 to 4006 (kBytes)           0         4050           2         3056           2         3056           2         3055           2         3055           2         3055           2         3055           2         3055           2         3055	MCUshaPUs L 	STM3: Low-P at: 2000 famms at: 2000 famms fittageon fittageon fittageon fittageon fittageon fittageon fittageon	2001.3 No ower & Stazool Com Stazool Com Stazool Adm Stazool Adm Stazool Adm Stazool Adm Stazool Adm Stazool Adm	ew Sub- Flexibili 0.000 NA 0.2013 0.000 NA 0.3335 0.000 0.553	IGHz So ity w.coP t2 w.coP t2 rssor49 rssor49 rssor49 rssor49 rssor40	2 483/643 32 483/643 35 483/643 36 483/643 36 483/643 36 483/643 36 483/643	6 kBytes 6 kBytes 6 kBytes 6 kBytes 6 kBytes 6 kBytes 6 kBytes	<b>577</b> 50 10 18 18 18 18 18	€ Expl 40 MHz 40 MHz
Package         2           Care         2           Dapocessor         3           MEMORY         4996           Primo 5 to 4006 (Abf/sec)         4996           0         15034           RM Total From 2 to 3026 (Abf/sec)         3026           RAM Total From 2 to 3026 (Abf/sec)         3026           RAM Total From 2 to 3026 (Abf/sec)         3036           CCM AAM From 5 to 302 (Abf/sec)         32           Oaklaps SIGAM From 5 to 100 (Abf/sec)         32	MCUMIPU, L	STM3: Low-P as: 200 term Maccon Maccon Maccon Maccon Maccon Maccon Maccon Maccon Maccon Maccon Maccon	2001.3 No ower & STA2201. Atm STA2201. Com STA2201. Com STA2201. Atm STA2201. Atm STA2201. Atm STA2201. Atm STA2201. Atm	ew Sub- Flexibili 000 NA 0.220 000 NA 0.3116 0.3116 0.3116 0.3116 0.3116 0.3116 0.3116 0.3116 0.3116 0.3116 0.3116	1GHz So ity MCCP 12 MCCP 12 MC	22 483/483 32 483/484 35 483/484 35 483/484 35 483/484 35 483/484 35 483/484 35 483/484 35 483/484	6 ABytes 6 ABytes 6 ADytes 6 ADytes 6 ADytes 6 ADytes 6 ADytes 6 ADytes 6 ADytes 6 ADytes 6 ADytes	2 ep 50 10 16 18 18 18 18 18 18	Expl 40 MPtc 40 MPtc
Package         2           Creat         2           Coprocessor         2           MEMORY         4956           EXPORT         4956           EXPORT         4956           EXPORT         50344           AM Trans 1 trans 0 to 15034 (Bbytes)         4956           AM Trans 1 trans 0 to 15034 (Bbytes)         50344           AM Trans 1 trans 0 to 15034 (Bbytes)         3205           AM Trans 1 to 3205 (Bbytes)         3205           Com Mark Trans 0 to 3205 (Bbytes)         32           Com Addres SIMM Trans 0 to 320 (Abytes)         32           B         709	NCUMPUL 1	STM3: Low-P 2012 2012 2014 102001 2014 2014 102001 2014 2014 102001 2014 2014 102001 2014 2014 102001 2014 2014 2014 102001 2014 2014 2014 102001 2014 2014 2014 2014 102001 2014 2014 2014 2014 102001 2014 2014 2014 2014 2014 102001 2014 2014 2014 2014 2014 102001 2014 2014 2014 2014 2014 102001 2014 2014 2014 2014 2014 2014 102001 2014 2014 2014 2014 2014 2014 201	Cover & Anno 2000 Control Cont	ew Sub- Flexibili teen NA 0.2213 soon NA 0.3116 0.3355 0.34566 0.345666 0.345666 0.345666 0.345666 0.345666666666666666666666666666666666666	IGHz So ity W.C9 II. 1500-8 1500-8 1500-8 1500-8 1500-8 1500-8 1500-8 1500-8 1500-8 1500-8	2 1946 2 245/45 32 45/45 32 45/45 35 45/45 36 45/45 36 45/45 36 45/45 36 45/45 32 45/45 32 45/45 32 45/45	6 kBytes 6 kBytes	<b>1</b> <b>10</b> 10 10 10 15 15 15 15 15 15 15 15 15 15	€ Expl 40 MP2 40 MP
Dackage         2           Core         2           Dopocessor         3           VEMORY         40%           Permon 3 to 40% (Adv/sec)         40%           PERPORM         Frain 0 to 15034 (Bdv/sec)           B         150.55           BM Total Frain 2 to 3005 (Bdv/sec)         3055           RAM Total Frain 2 to 3005 (Bdv/sec)         3025           CCM MARK Frain 0 to 300 (bdv/sec)         325           CCM MARK Frain 0 to 300 (bdv/sec)         32           Ob adapt SBARM Frain 0 to 300 (bdv/sec)         32           Ob adapt SBARM Frain 0 to 700 (bdv/sec)         32           Ob adapt SBARM Frain 0 to 700 (bdv/sec)         32           Ob adapt SBARM Frain 0 to 700 (bdv/sec)         32	NCUMPON L	STM3: Low-P is: 3000 terms incocore incoco	2WL3 No ower & station composite station composi	ew Sub- Flexibili 0.2000 0.200 0.200 0.200 0.200 0.200 0.200 0.200 0.200 0.200 0.200 0.200 0.200 0.200 0.200 0.200 0.2000 0.2000 0.2000 0.2000 0.2000 0.2000 0.2000 0.2000 0.2000 0.2000 0.2000 0.2000 0.2000 0.2000 0.200000000	IGH2 So ity W.COP U. W.COP U. W.COP U. TSDO-49 TSDO-49 TSDO-49 TSDO-49 TSDO-49 TSDO-49 TSDO-49 TSDO-49 TSDO-49 TSDO-49 TSDO-49 TSDO-49	C Fands 2 22 Hillyses 32 Hillyses 35 Hillyses 36 Hillyses 36 Hillyses 36 Hillyses 36 Hillyses 36 Hillyses 36 Hillyses 36 Hillyses 36 Hillyses 32 Hillyses 32 Hillyses 32 Hillyses	6 403/000 6 403/000	<ul> <li>P2</li> <li>P3</li> <li>P4</li> <li>P</li></ul>	Expl 40 MPtc 40 MPtc
Package         2           Care         2           Copocessor         2           Copocessor         2           MEMORY         40%           EXPONE         40%           EXPONE         40%           EXPONE         5034           Mail Total         5034           SMI Total         700           SMI Total	MCUMBUL L 	STM3: Low-P 2020 terms Maccon Maccon Maccon Maccon Maccon Maccon Maccon Maccon Maccon Maccon Maccon Maccon Maccon Maccon Maccon Maccon Maccon	2000 2 20	<b>Superior Sub-</b> <b>Flexibili</b> <b>seen</b> NA <b>0.223</b> <b>seen</b> NA <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3355</b> <b>0.3555</b> <b>0.3555</b> <b>0.35</b>	IGHZ So ity	2 483/64 32 483/64 32 483/64 35 483/64 16 483/64 16 483/64 16 483/64 16 483/64 16 483/64 16 483/64 18 483/64 12 483/64 12 483/64 12 483/64 12 483/64	6 kBytes 6 kBytes	<ul> <li>10</li> <li>1</li></ul>	€ Expl 44 MHz 48 MHz
Package 2 Com	MCUMPUA U - 00 -	STIM3: LOW-P JU 2020 Berry Incodes Inc	WILS NO	ew Sub- Flexibili son M 8.223 son M 8.223 son M 8.235 son M 8.235 son M 8.235 son M 8.235 son M 8.235 son M 8.243 son M 8.244 son M 8.2444 son M 8.2	IGHz So ity W.CP 12. W.CP 12. W.CP 12. W.CP 12. W.CP 12. U.CP 12.	2 Family 22 Highes 23 Highes 24 Highes 24 Highes 26 Highes 26 Highes 26 Highes 20 Highes 22 High	6 kBytes 6 kBytes	<ul> <li>P2</li> <li>P2</li> <li>P3</li> <li>P4</li> <li>P</li></ul>	Expl 44 MPz 48 M
Package         2           Care         2           Coprocessor         2           MEMORY         40%           EXPOSE         40%           EXPOSE         40%           EXPOSE         50%           State         50%           St	MCUstarUs L C 3 C 3 C 3 C 3 C 3 C 3 C 3 C 3	STIM3: LOW-P 2012 2020 forms 2012 2020 for any 2012 2020 for any 20	20013 No. 2014 No. 20	Sub-     Flexibil:     Source     Sub-	1GHz So ity	C 2483/64 22483/64 25483/64 25483/64 25483/64 25483/64 25483/64 25483/64 25483/64 25483/64 25483/64 25483/64 25483/64 25483/64 25483/64 25483/64 25483/64 25483/64 25483/64		2 23 70 70 70 70 70 70 70 70 70 70 70 70 70	Expl 40 MPa 40 MPa
PACKapt 2 Com		STIM3: LOW-P Integent	WILS Nu owers & Macon Anno Macon	www.sub- Flexibil:     wowness     sub- su	LGHz So ity WLCP 12 WLCP 12 TSOA 13 TSOA 13 TS	22 483/105 22 483/105 25 483/105 25 483/105 26 483/105 26 483/105 27 483/105 22 483/105 23 483/105 24 483/105	6 kBytes 6 kBytes	2 20 30 10 38 38 38 38 38 38 38 38 38 38 38 38 38	Equition 1 = 1 = 1 = 1 = 1 = 1 = 1 = 1 = 1 = 1

Figure 7: Target Selection Window





#### Select the MCU

In the "Target Selection" window, look for the box labeled "Commercial Part Number". This is where the part number of the MCU on the STM32H747I-DISCO will be entered. Enter "STM32H747XI" and select the presented device.

MCU/MPU Selector	Board Selector	Example Selector	Cross Selector
MCU/MPU Filters	STM32H7477 STM32H7477	S XIH6	Enter the Part Number here
Segment		>	

Figure 8: Select the STM32H747XIH6 Device

Once the MCU has been selected then click on the device in the "Features" section as outlined in blue.

CU/MPU Filters	3	Features Block D	liagram Docs & Resources CAD	) Resources 📑 Datasheet 🏹 Buy
Commercial STM32H74	17XIH6 ~	STM32H7 Series	×	
		STM32H747XIH6	High-performance and DSP with DP-FP 2MBytes of Flash memory, 1MB RAM, 4 external memory interface, large set of	U, Arm Cortex-M7 + Cortex-M4 MCU with 80 MHz CPU, Art Accelerator, L1 cache, peripherals, SMPS
PRODUCT INFO	evice 🗸	ACTIVE	Unit Price for 10kU (US\$): 11.5667	
Segment	<u> </u>	Product is in mass production	Boards: STM32H747I-DISC1 - STM32H747I-DISCO - STM32H747I-EVAL	TFBGA 240+25 14x14x1.2 P 0.8 mm
Series				
Line		STM32H747xI/G devices are ba operates at up to 480 MHz and	sed on the high-performance Arm <sup>®</sup> Cortex <sup>®</sup> -M7 and the Cortex <sup>®</sup> -M4 core at up to 240 MHz. Both cores fea	Cortex <sup>®</sup> -M4 32-bit RISC cores. The Cortex <sup>®</sup> -M7 core ature a floating point unit (FPU) which supports Arm <sup>®</sup>
Marketing Status	X	single- and double-precision (C	ortex®-M7 core) operations and conversions (IEEE 75	54 compliant), including a full set of DSP instructions and a
Price		STM32H747xI/G devices incorp	orate high-speed embedded memories with a dual-ba	ank flash memory of up to 2 Mbytes, up to 1 Mbyte of RAM
Раскаде	2	(including 192 Kbytes of TCM R enhanced I/Os and peripherals	AM, up to 864 Kbytes of user SRAM and 4 Kbytes of connected to APB buses, AHB buses, 2x32-bit multi-A	backup SRAM), as well as an extensive range of AHB bus matrix and a multi layer AXI interconnect
Core	~	supporting internal and externa	I memory access.	ower RTC, a high-resolution timer, 12 general-numose
Coprocessor		16-bit timers, two PWM timers for	or motor control, five low-power timers, a true random	number generator (RNG). The devices support four digital
MEMORY	~	inters for external signa-deita in	odulators (Dr. SUM). They also realtire standard and	advanced communication menaces.
Flash = 2048 (kBytes)		Features		
2048		Includes ST state-of-the	e-art patented technology	
EEPROM = 0 (Bytes)		Dual core		
0				
RAM Total = 1024 (kBytes)		MCUs/MPUs List: 1 item		🖒 Expo
1024		Commer Part No	Reference Marketing X Unit Price L X Board X	Package X Flash X RAM X I/O X Freque
RAM = 1024 (kBytes)		☆ S1M32H747STM32H747X	S1M32H74 Active 11.5667	TFBGA 240 2048 kBytes 1024 kBytes 172 480 MHz
1024		1		
CCM RAM = 0 (kBytes)				
0		Click here to Select Device	E	

Figure 9: Selecting the STM32H747XIH6 Device in Features Section





Then click "Next" as highlighted in green.

MCUs/MPUs List: 1 item					🕂 Export
Commer * ☆ STM32H747	Part No Reference   STM32H747XI STM32H74 /	Marketing × Unit Price f Active 11.5667	X Board X Package ST ST ST TFBGA 240	× Flash × RAM × 2048 kBytes 1024 kBytes	I/O         ×         Frequency         ×           172         480 MHz
Click here to Sele	ct Device				
				Click "Next" to Co	ntinue

Figure 10: Select "Next" to Continue Project Creation

### Name the Project

There are several steps left to finish the project creation. Naming the project is the next step. Keep the default settings as shown in the image below.

Project		
Project Name:	stm32h747_e35rd-mw420-c	
🕝 Use default	location	
Location:	C:/Users/tonys/STM32CubelDE/workspace_1.14.1	Browse
Targeted Bi	nary Type O Static Library	
Targeted Pr STM32C	oject Type ube 🔿 Empty	

Figure 11: Naming the Project





After naming the project, select "Next" to verify the included libraries.

DE STM32 Project		-		×
Firmware Library Package Setu	μ			DE
Setup STM32 target's firmware				DE
Target and Firmware Package				
Target Reference:	STM32H747XIHx			
Firmware Package Name and Version:	STM32Cube FW_H7	V1.11.1	~	
Firmware and Software Package Repo	sitory			
Location:				
C:\Users\tonys\SIM32Cube\Reposito	ry			
See <u>'Firmware Updater'</u> for settings re	lated to package inst	allation		
Code Generator Options				
$\bigcirc$ Add necessary library files as refere	nce in the toolchain p	roject conf	figuration	n file
Copy all used libraries into the proj	ect folder			
Copy only the necessary library file	s			
(?) < Back N	ext > Finis	ih	Cance	el
Eiguro 12	· Library Dackage	Cotup		

Figure 12: Library Package Setup

Verify that only the necessary libraries are included.

The next two pop up dialogs have optional selections. In both cases, "Yes" was selected. The first is whether the Pinout Configuration Perspective is opened after project generation is completed. "No" can be selected with no effect on the code base.



*Figure 13: Open the Pin Configuration Perspective* 

The second pop up dialog will influence code generation. The dialog asks if it should enable the Memory Protection Unit (MPU) to optimize the Speculative Read access of the M7 core. This will add code for configuring the registers involved with the MPU and Speculative Read. This is a minor amount of code and should not affect the operation of the display. Again, selecting "No" is a viable





option but in the case of this application note "Yes" was selected and the resulting code compiled into the firmware.



Figure 14: Memory Protection Unit Enable

Once those steps are completed the STM32CubeIDE will display:



Figure 15: STM32CubeIDE Pin Perspective





### Add the Files to the Project

The Pinout Configuration perspective can be closed, and the C/C++ perspective opened.

File Life Source Referent Navigete Service Project flue Winds 177 • 16 10   10 • 5 • 16   17   18   18 • 18 • 18														-	o x
12-11010-2-10101x1010-0-10	m Help 1 Hele Tany														
	· · · · · · · · · · · · · · · · · · ·	81位+日+中	40.0.16	0										٩,	न छ 🗖
Prove trave x		12+5+0	<b>₩</b> • • • •   <b>1</b>	0							• •	Contract X (B)	hair begin	4	
	B Publics × 20 Table © Console. ○ B Revel Description	Srgetes Researce	Fats	Location	See.	71-0	in hold Andyse Memory Region Region	X <u>ak</u> Turk Stack Marwoy Datain Dart address	Analyser @Cy End addess	Ownanic Comple Sat	ing .	Ureaf -	UsgeTO		e I • 0

Figure 16: C/C++ Perspective

The next step in the creation of the project is to add in the header and source files provided by Focus LCDs and read FAN4222 and FAN4223. Many of the files used throughout these projects are the same. The only difference is the few files based on the differences in the display initialization and timing. Those modifications will be presented in the section: **Creating the Firmware Files for the E35RD-MW420-C**.

The complete project files for the **E35RD-MW420-C**, requiring no modification, can be provided by Focus LCDs upon request. This application note is meant as a guide to developing firmware for a display without completed project files.

First, add the include files to the project by first opening the development environment workspace and navigating to the CM7 Inc folder. As an example: **.../stm32h747\_e35rd-mw420-c/CM7/core/Inc**. There will be 3 initial files, generated by the tools, in the folder.

clock.h	debug.h	delay.h	dsi.h
fmc.h	focusimage.h	ft5x16.h	fwver.h
gui.h	i2cs.h	lcd.h	mcuid.h
misc.h	touch.h	tpc_descriptors.h	usart.h

Below is the list of header files to copy into the workspace Inc folder.





Once the files are copied over to the proper workspace folder location, Project Explorer in the IDE will need to be refreshed to import the added files.

Right click on: **stm32h747\_e35rd-mw420-c\_CM7** to bring up the menu and select refresh. Once the refresh has been completed the newly added files will be part of the project.



Figure 17: Project Explorer before Refresh



Figure 19: Refresh of Inc Folder





The same procedure will be used to add the necessary files to the source folder. The files will be placed in **.../workspace\_1.14.1/stm32h747\_e35rd-mw420-c/CM7/core/Src**. There are 5 files, generated by the dev tool, already in that folder. The files that will be added are.

clock.c	debug.c	delay.c	dsi.c
fmc.c	focusimage.c	ft5x16.c	fwver.c
gui.c	i2cs.c	lcd.c	misc.c
touch.h	usart.h		

Again, the project will need to be refreshed. Right click on: **stm32h747\_e35rd-mw420-c\_CM7** to bring up the menu and select refresh. Once the refresh has been completed the newly added files will be part of the project as shown below:



Figure 20: Refresh the Src Folder





#### Modify main.h and main.c

The header files must be included into the main.h file so that the resulting code can be compiled into the firmware. Insert the header files in the code section marked by: /\* USER CODE BEGIN Include \*/. The debug macro can be placed in the section marked by: /\* USER CODE BEGIN EM \*/. Modifications to main.h are shown in the image below:

750 / * * = 1 = 1 = -	****************
76 ^ Includes	******
70 <b>#inglude</b> "ctm22b7ww bol b"	
78 #Include "SUMSZN/XX_Nal.N"	
900 /* Privato includos	*/
81 /* USEP CODE REGIN Includes */	
82 <b>#include</b> "mcuid b"	
83 <b>#include</b> "clock b"	
84 <b>#include</b> "delay.h"	
85 <b>#include</b> "debug.h"	
86	
87 <b>#include</b> "fmc.h"	
88 <b>#include</b> "dsi.h"	
89	
90 #include "touch.h"	
91 #include "gui.h"	
92	
93 /* USER CODE END Includes */	
94	
950/* Exported types	*/
96 /* USER CODE BEGIN ET */	
97	
98 /* USER CODE END ET */	
99	
1000/* Exported constants	*/
101 /* USER CODE BEGIN EC */	
102	
103 /* USER CODE END EC */	
104	
1050/* Exported macro	*/
106 /* USER CODE BEGIN EM */	
107 #define debugMAIN(type,)	<pre>printDEBUG(type, "SYS",VA_ARGS)</pre>
108 / ^ USER CODE END EM ^/	

Figure 21: Main.h Modifications

# FAN4224



Next, main.c will require some additions for the application code. Looking at the image below, in the first green boxed area, the peripheral and subsystem initialization functions were added under the section: /\* USER CODE BEGIN 2 \*/. The first few lines of code enable User LED1 on the STM32H747 Discovery board. LED1 will blink at a 1Hz rate.

In the same section the macro **initDEBUG()** and **debugMAIN()** are added to allow debug messages to be sent over the USART (Universal Synchronous/Asynchronous Receiver Transmitter).

Finally, the **chkGUI()** function is called in the **while(1)** loop to respond to touch events. The chk4TimeoutSYSTIM() is used to create a 1 Hz timeout so that use LED1 can be blinked once a second. This verifies the application is running.

```
h e35rd_mw420_c.h c e35rd_mw420_c.c h main.h
                                 .c main.c ×
178
       // Verify the MCU has started correctly by turning on LED1
179
      RCC->AHB4ENR |= RCC AHB4ENR GPIOIEN;
180
       GPIOI->MODER &= ~ (GPIO MODER MODE12);
181
       GPIOI->MODER |= (GPIO_MODER_MODE12 0);
182
       GPIOI->ODR |= 0x1000;
183
184
       // Enable the I and D cache
185
       SCB EnableICache();
186
       SCB EnableDCache();
187
188
       initCLOCK();
189
       initSYSTIM();
190
191
       initDEBUG("[CM7]", '4', "STM32H747 MIPI Display Test");
192
       initMICTIM();
193
194
       initFMC();
195
196
       // This calls initTouch() and initDSI()
197
       initGUI();
198
       debugMAIN(0, "init done\n");
199
200
201
       uint32 t timer = getSYSTIM();
202
         USER CODE END 2 */
203
204
       /* Infinite loop */
205
       /* USER CODE BEGIN WHILE */
206
       while (1)
207
208
         chkGUI();
209
210
         if(chk4TimeoutSYSTIM(timer, 1000) == (SYSTIM TIMEOUT))
211
         {
212
             timer = getSYSTIM();
213
             GPIOI->ODR ^= 0x1000;
214
215
         /* USER CODE END WHILE */
216
217
         /* USER CODE BEGIN 3 */
218
       }
```







### **Change Optimization Settings**

The next step in creating the firmware is setting the optimization level. The typical setting is '-O2' for a good compromise between execution speed and code size. If left at '-O0' there would be no optimization for execution speed and code size.

First, in Project Explorer right click on the (CM7) project and select properties.

pe stm32h747_e35rd-mw420-c_CM7	(in CM7	n	138 init
> 🐉 Binaries		New	>
> D Includes		Go Into	
> 🔄 Common		o : N	
V 📇 Core		Open in New Window	
> 🔁 Inc		Show In	Alt+Shift+W >
V 🕞 Src		Conv	Ctrl+C
> [c] clock.c		сору	Cul+C
> c debug.c		Paste	Ctrl+V
> c delay.c	×	Delete	Delete
> c dsi.c		Source	>
> c e43gb_1_mw405.c		Move	
> ic mc.c		Pename	E2
> To fuerer c		Kename	12
> C iWver.c	20	Import	
> C main c	2/2	Export	
> C misc c	lhoudi		
stm32h7xx hal msp.c		Build Project	
> stm32h7xx it.c		Clean Project	
> C syscalls.c	S	Refresh	F5
> c sysmem.c	-	Close Broject	
> c touch.c		close Project	
> c usart.c		Close Unrelated Projects	
> 😝 Startup		Build Configurations	>
> 🚑 Drivers		Build Targets	
> 🔁 Debug		build largets	
STM32H747XIHX_FLASH.Id		Index	>
STM32H747XIHX_RAM.Id	0	Run As	>
stm32h/4/_e35rd-mw420-c.ioc	林	Debug As	>
		Team	>
		Compare With	
			,
	1	Restore from Local History	
		Generate Code	
	C &	Convert to C++	
	**	Run C/C++ Code Analysis	
		Configure	>

Figure 23: Project Properties Main Menu Item





In the pop-up window, select C/C++ Build->Settings.

type filter text	Settings		↓ ↓ ↓
<ul> <li>Resource</li> <li>Builders</li> <li>C/C++ Build</li> <li>Build Variables</li> </ul>	Configuration: Debug [ Active ]	~	Manage Configurations.
Environment Legging Settings C/C++ General CMSIS-SVD Settings Project References Run/Debug Settings	Tool Settings Puild Steps	Build Artifact Binary Parsers Serror I Select what toolchain to use Use workspace setting (GNU Tools for S Fixed GNU Tools for STM32 (11.3.ref)	Parsers TM32 (11.3.rel1))
	General Debugging Preprocessor	Toolchain Manager Configure the workspace toolchain, and m	nanage installed toolchains.
	Miscellaneous	Open Toolchain Manager	

*Figure 24: C/C++ Build Settings* 

Then in the Tool Settings tab on the right, select Optimization and set the Optimization level to -O2.



Figure 25: Optimization Level

#### **Build the Project**

The final step to setting up the project is to build the project. This step ensures that there are no missing dependencies. If there are any errors or warnings, resolve those issues before continuing with the firmware project.





## Creating the Firmware Files for the E35RD-MW420-C

Using the E43GB-I-MW405-C firmware files as a reference, the E35RD-MW420-C display will be discussed in this section. This will guide the end user through the necessary steps of creating the firmware for the display.

These are the basic steps to creating the firmware:

- 1. Create the display specific files e35rd\_mw420\_c.c/.h.
- 2. Edit the header file with the inclusion guard, include headers, and API
- 3. Create the display controller header file st7701s.h.
- 4. Edit the header file to add the #define for the ST7701S registers.
- 5. Edit the e35rd\_mw420\_c.c file with the **#include** header file names.
- 6. Check the LCD controller chip datasheet, the Focus LCDs initialization code, and continue to modify the source file as necessary.
- 7. Edit the .c file with the new display parameters.
- 8. Make the initialization sequence changes to runInitSeqLCDConfig() function.

These steps will be discussed in the following sections.

#### Create the E35RD-MW420-C Header and Source Files

Using the e43gb\_i\_mw405\_c.h and .c files as examples, create a new header file in the Inc folder and a new source file in the Src folder. Name the files e35rd\_mw420\_c.h and e35rd\_mw420\_c.c. Once that is completed, code will be added to the new header file.







#### Editing the Header File

In the header file, verify that the **#ifndef E35RD\_MW420\_C\_H** matches the new header file name.

#ifndef	E35RD	MW420	С	Η
#define	E35RD	MW420	С	Η

Figure 28: Include Guard

Add the include preprocessor commands for the standard C library headers and for the project headers. The <stdarg> header is needed for the \_\_VA\_\_ARGS\_\_ used in the debug code inside the source file.

880	/******	* * * * * * * * * * * * * * * * * * * *
89	* Includ	les
90	******	*****
91	#include	<stdint.h></stdint.h>
92	#include	<stdarg.h></stdarg.h>
93	#include	"mcuid.h"
94	#include	"debug.h"
95	#include	"delay.h"
96	#include	"dsi.h"
97	#include	"st7701s.h"

Figure 29: Include Headers

At this point defines could be added for the LCD width and height in pixels to help make the code more portable between displays. Once that is complete the function prototypes for the API can be placed in the file.

Figure 30: API





#### Create and Edit the ST7701S Header File

In the header file, verify that the **#ifndef ST7701S\_REG\_H** matches the new header file name.

65	#ifndef	ST7701S_	REG_H	ł
66	#define	ST7701S	REG_H	ł

Figure 31: ST7701S Header File Include Guard

Looking at the ST7701S datasheet, the section on the commands/registers is what will populate the header file. Start by #define the commands and data bytes for changing register pages.

h e35rd_mw	420_c.h	c e35rd_mw420_c.c	h main.h	c main.c	h st7701s.h	X
779/*	*****	* * * * * * * * * * * *	* * * * * * * * * *	*****	* * * * * * *	* * * * * * * * * * * * * * * * * * * *
78 *	Globa	l Constants,	/Defines			
79 *	*****	* * * * * * * * * * * * *	* * * * * * * * * *	* * * * * * *	* * * * * * *	*****
80 /*	** E1	nable Extend	ded Comman	d Regis	ters -	has 5 parameters ***********/
81 <b>#d</b>	lefine :	ST7701S_CN2B	BKSEL	0xFF	11	Enable Command Bank Select register
82 <b>#d</b>	lefine :	ST7701S_CN2	BKSEL_PARA	M1 0x77	11	Parameter 1 - required
83 <b>#d</b>	lefine :	ST7701S_CN2B	BKSEL_PARA	M2 0x01	11	Parameter 2 - required
84 <b>#d</b>	lefine :	ST7701S_CN21	BKSEL_PARA	M3 0x00	11	Parameter 3 - required
85 <b>#d</b>	lefine :	ST7701S_CN2	BKSEL_PARA	M4 0x00	11	Parameter 4 - required
86					11	Parameter 5 - use only 1 option below
87 <b>#d</b>	lefine :	ST7701S_CMD	1	0x00	11	Command 1 register set
88 <b>#d</b>	lefine :	ST7701S_CMD2	2_BNK0	0x10	11	Command 2 Bank 0 register set
89 <b>#d</b>	lefine :	ST7701S_CMD2	2_BNK1	0x11	11	Command 2 Bank 1 register set
90 <b>#d</b>	lefine :	ST7701S_CMD2	2_BNK3	0x13	11	Command 2 Bank 3 register set

Figure 32: Page/Bank Change Command and Data

Next, add the rest of the command registers to the header file.

h e35rd	_mw420_c.h	e35rd_mw420_c.c h main.h	C main.c	▶ st7701s.h ×
93	/******	****** Command 1 Ba	ink 0	***************************************
94	#define	ST7701S_NOP	0x00	) // no operation
95	#define	ST7701S_SWRESET	0x01	// software reset
96	#define	ST7701S_RDDID	0x04	<pre>// read device ID (returns 3 bytes)</pre>
97	#define	ST7701S_RDNUMED	0x05	// read # of DSI errors
98	#define	ST7701S_RDRED	0x06	5 // read first pixel of color R
99	#define	ST7701S_RDGREEN	0x07	// read first pixel of color G
100	#define	ST7701S_RDBLUE	0x08	B // read first pixel of color B
101	#define	ST7701S_RDDPM	0x0A	A // read display power mode
102	#define	ST7701S_RDDMADCTL	0x0B	3 // read display MADCTL

Figure 33: Register Defines





Some command registers have named bit fields. Define these bit names so they can later be used to configure their associated register. See the COLMOD register and bit defines below for an example.

h e35rd_mw420_c.h	c e35rd_mw420_c.c h main.h	💽 main.c 🔥 st7	:t7701s.h 🗙
140 /******	***** Command 1	COLMOD *****	***************************************
141 #define	ST7701S_COLMOD	0x3A	<pre>// interface pixel format</pre>
142 #define	BPP16	0x50	// 16 bits per pixel format: 65k color
143 #define	BPP18	0x60	// 18 bits per pixel format: 262k color
144 #define	BPP24	0x70	// 24 bits per pixel format: 16.7M color
		5:	
143 <b>#define</b> 144 <b>#define</b>	BPP18 BPP24	0x60 0x70	<pre>// 18 bits per pixel format: 262k color // 24 bits per pixel format: 16.7M color</pre>

Complete all the register and bit defines for the ST7701S.

#### Editing the Source File

First, looking at the datasheet for the E35RD-MW420-C, notice the TFT controller chip is the ST7701S.

In the e35rd\_mw420\_c.c source file, the #include directives must be added for E35RD-MW420-C and the ST7701S. The header file include should be **#include "e35rd\_mw420\_c.h"** and **#include "st7701s"**.

The debugging system requires a constant module name and a macro that provides an alias to the function pointer for printing debug messages.



#### Figure 35: Constant Name and Macro Alias

#### Setting up Arrays for the Command Paramters

Many of the commands for controlling the initialization of the display have associated parameters. One of the first items to code are the arrays with the data for the commands as the DSI packet write functions (provided by the ST include and source files generated with project creation) require arrays when there is more than 1 byte of data for a command.





Different display controllers have command sequences to access additional register pages or banks.

```
h e35rd_mw420_c.h c e35rd_mw420_c.c × h st7701s.h h main.h c main.c
106 * Private Variables
    107
1080// Page/Bank change to access the registers on different pages/banks
109 // 0x77, 0x01, 0x00, 0x00, 0x00
110 uint8_t Page0[] = {ST7701S_CN2BKSEL_PARAM1, ST7701S_CN2BKSEL_PARAM2,
                    ST7701S CN2BKSEL PARAM3, ST7701S CN2BKSEL PARAM4,
111
112
                    ST7701S CMD1};
113 // 0x77, 0x01, 0x00, 0x00, 0x10
114 uint8 t Page1[] = {ST7701S CN2BKSEL PARAM1, ST7701S CN2BKSEL PARAM2,
                    ST7701S CN2BKSEL PARAM3, ST7701S CN2BKSEL PARAM4,
115
                    ST7701S_CMD2_BNK0};
116
117 // 0x77, 0x01, 0x00, 0x00, 0x11
118 uint8 t Page2[] = {ST7701S CN2BKSEL PARAM1, ST7701S CN2BKSEL PARAM2,
119
                    ST7701S CN2BKSEL PARAM3, ST7701S CN2BKSEL PARAM4,
120
                    ST7701S CMD2 BNK1};
121 // 0x77, 0x01, 0x00, 0x00, 0x13
122 uint8 t Page3[] = {ST7701S_CN2BKSEL_PARAM1, ST7701S_CN2BKSEL_PARAM2,
123
                    ST7701S CN2BKSEL PARAM3, ST7701S CN2BKSEL PARAM4,
124
                    ST7701S CMD2 BNK3};
```

Figure 36: Arrays for Command Pages

#### **Initial Parameter Configuration**

The parameters of the TFT LCD will be set and initialized in the void **initLCDConfig**(LCDCONF **\*lcdconfig**) function in the e35rd\_mw420\_c.c file. The argument **LCDCONF** is a structure containing all the essential information required by the DSI controller to configure the MIPI DSI interface.

#### Command or Video Mode and Additional Configuration

Identifying whether the display operates in MIPI video or command mode can be verified by reviewing the controller datasheet for GRAM. If the controller has GRAM, it can operate in command mode but if there is no GRAM it must operate in video mode. Reviewing the ST7701S datasheet shows that it has no internal GRAM and operates in video mode. In the source file the mode is set to **DSI\_VIDEO\_MODE** and does not need to be changed as both displays use the same controller.









Looking at the datasheet, the display resolution is 480 x 800. Modify the configuration height and width. The **LCD\_WIDTH** and **LCD\_HEIGHT** are defined in the e35rd\_mw420\_c.h file.

_mw420_c.h	€ e35rd_mw420_c.c ×	h st7701s.h	h main.h	🔂 main.c	
lcd	lconf->width =	LCD_WID	TH;		
lcd	lconf->height	= LCD_HE	IGHT;		

Figure 38: Adjusted Height and Width

The next set of parameters to consider are the display timings. These can sometimes be found in the display datasheet or from the comments section of the display initialization file. Both can be found on the Focus LCDs website or are available upon request. Carry the over the display timings to the LCDCONF structure.

```
NOTE:VCI=3.3V,IOVCC=1.8V,
Display resolution:480*800
params->dsi.vertical_sync_active=2
params->dsi.vertical_backporch=20
params->dsi.vertical_frontporch=10
params->dsi.horizontal_sync_active=2
params->dsi.horizontal_backporch=60
params->dsi.horizontal_frontporch=10
params->dsi.PLL_CLOCK=(350)Mbps
params->dsi.PLL_CLOCK=(175)Mbps //FOR MTK
params->RGB_CLOCK=(28)MHZ
Frame Rate=60HZ
```

Figure 39: E35RD-MW420-C Display Timings from the Display Initialization File

```
w420_c.h 💽 e35rd_mw420_c.c 🗙 庙 st7701s.h 庙 main.h 💽 main.c 💽 e35rd_mw420_c.c
  lcdconf->hact = lcdconf->width;
                                               // horizontal address
                                               // vertical address
  lcdconf->vact = lcdconf->height;
  lcdconf->hsw = 4;
                                               // horizontal synchronization width
                                                // horizontal back porch
  lcdconf -> hbp = 80;
                                                // horizontal front porch
  lcdconf -> hfp = 20;
                                                // vertical synchronization height
  lcdconf -> vsh = 4;
                                                // vertical back porch
  lcdconf->vbp = 20;
  lcdconf->vfp = 10;
                                                // vertical front porch
                                               // horizontal active area
  lcdconf->haa = lcdconf->width;
                                                // vertical active area
  lcdconf->vaa = lcdconf->height;
                                                // horizontal synchronization active
  lcdconf->hsa = lcdconf->hsw;
  lcdconf->vsa = lcdconf->vsh;
                                                // vertical synchronization active
```

Figure 40: Display Timings





Due to slight variations in timing, some parameters may need to be adjusted to ensure proper functioning of the display.

#### **Configuring the Initialization Sequence**

Having adjusted the timings for the display, the initialization sequence will be modified for the new display. In the previous section, the display timings were retrieved from the display initialization file. The commands for display initialization are included in that file.

```
//************************/
write_command(0xFF);
write_data(0x77);
write_data(0x01);
write_data(0x00);
write_data(0x00);
write_data(0x00);
write_data(0x13);
```

Figure 41: Switch Command Pages

There are two types of writes, command and data. The write\_command() is the address of the register where the data will be written. The write\_data() is the data that will be written to the register.

Commands that take more than 1 parameter require the use of the long write function, example above longWriteDSI(lcdconf->virch, long\_pkt\_write\_type, 5, 0xFF, Page3). Commands that require 1 or no data parameters are call short write functions like shortWriteDSI(lcdconf->virch, short\_pkt\_write\_type, 0xEF, 0x08). In the initialization code when a command has no corresponding data then a short write is used with the last parameter being set to 0x00.



Figure 42: Display Initialization Long and Short Writes





The display initialization code is generic 'C' code to show the concept of registers and data to be placed in those registers. This is where the MIPI DSI concepts of short and long write packets are used for data transfer. The end user needs to convert the display initialization code into short and long DSI packet writes. The function used for calling all the initialization code is



Continue to edit the void runInitSeqLCDConfig(LCDCONF \*lcdconf) function. This is required for all the write commands and associated data in the display initialization file. Shown below is a section of the additions to runInitSeqLCDConfig() function.

20_c.h	€ e35rd_mw420_c.c × 🕞 st7701s.h 🕞 ma	in.h 🖻 main.c 💽 e35rd_mw420_c.	c				
sh	ortWriteDSI(lcdconf->virch,	short pkt write type,	0xB0,	0x5D);	// V	op	
sh	ortWriteDSI(lcdconf->virch,	short pkt write type,	0xB1,	0x6D);	// V	COM = 5A	
sh	ortWriteDSI(lcdconf->virch,	short pkt write type,	0xB2,	0x87);	// V	GH = 15V	
sh	ortWriteDSI(lcdconf->virch,	short_pkt_write_type,	0xB3,	0x80);	// T	est Command setting	
sh	ortWriteDSI(lcdconf->virch,	short_pkt_write_type,	0xB5,	0x49);	// V	GL = -12V	
sh	ortWriteDSI(lcdconf->virch,	short pkt write type,	0xB7,	0x85);	// P	ower Control 1	
Figure 44: A Sample of the runInitSeqLCDConfig() Function							

In the figure above, the **shortWriteDSI()** functions are used to send commands to the display. The short packet functions are used when a command with no data or 1 data parameter needs to be transmitted to the display. Long packet writes are used when there is more than 1 data parameter to transmit.

Looking at the display initialization code, there are 5 parameters to the page change command. In the figure below, an example of a long write packet is the DSI version of the page change command. The last parameter, **Page3**, is the array which stores the **5** parameters.



# FAN4224



The commands that have several data parameters should have arrays for the data parameters to be passed to the **longWriteDSI()** function. Some of these commands include the Positive and Negative Gamma settings and the GIP settings. Once all the changes have been made to the various headers and source files, the demo will build and can be downloaded onto the STM32H7 Discovery board with the appropriate display attached.



Figure 46: E35RD-MW420-C Running the Focus LCDs Demo





## **Additional Comments**

The source code with the modifications already completed with the addition of the required code for driving the touch panel can be provided by Focus LCDs upon request.

## Summary

In Part 1, the hardware requirements for a MIPI display demo were presented. How to assemble the hardware was shown. Finally, a brief overview of the MIPI DSI interface and the STM32 DSI Host peripheral were discussed.

In Part 2 presented here, the development tools and frame buffer consideration were briefly discussed. The demonstration firmware operation was mentioned along with the touch interface. The following sections went through the operation of modifying the firmware for the E50RA-I-MW490-C display. Through the modification of the firmware, the structure and basic layout of the code is shown.

# FAN4224



## LCD Handling Precautions

- Do not store the TFT-LCD module in direct sunlight, best stored in a dark place
- Do not leave it exposed to high temperature and high humidity for a long period of time
- Recommended temperature range is 0 to 35 °C, relative humidity should be less than 70%
- Stored modules away from condensation as formation of dewdrops may cause an abnormal operation or failure of the module.
- Protect the module from static discharge
- Do not press or scratch the surface and protect it from physical shock or any force

## Disclaimer

Buyers and others who are developing systems that incorporate FocusLCDs products (collectively, "Designers") understand and agree that Designers remain responsible for using their independent analysis, evaluation, and judgment in designing their applications and that Designers have full and exclusive responsibility to assure the safety of Designers' applications and compliance of their applications (and of all FocusLCDs products used in or for Designers' applications) with all applicable regulations, laws, and other applicable requirements.

Designer represents that, with respect to their applications, Designer has all the necessary expertise to create and implement safeguards that:

(1) anticipate dangerous consequences of failures

(2) monitor failures and their consequences, and

(3) lessen the likelihood of failures that might cause harm and take appropriate actions.

The designer agrees that prior to using or distributing any applications that include FocusLCDs products, the Designer will thoroughly test such applications and the functionality of such FocusLCDs products as used in such applications.

### **Revision History**

Revision	Notes	Date
1.0.0	Initial Version	8/27/2024
1.1.0	Added Additional Code Sections for Clarification	9/6/2024
1.2.0	Added Running Demo Images	9/8/2024