ProtogenSmartController

Mark-II

ProtoOpen Development Manual

Document V1.0
Applied to ProtoOpen V1.0

MangMuang's Elektronik
2023

Document History

Version	Date	Description
1.0	08/10/2023	Initial

About this manual

- This manual was made for guiding how to develop our own code to upload, compile and execute on your Protogen
- For your safety, please read this manual before coding your Protogen.
- Information may change without further notice.
- If you have any questions, you may contact the manufacturer for help.

Manual Coverage

Which Protogen does this manual covered.

Controller Core Structure Type

- ☐ Standard Core MK-I Plus (None OS)
- ☐ Standard Core MK-II (RTOS Based)
- ☐ Extended Core MK-II (AI OS Based)
- ☑ Developer Core MK-II (AI OS Based)

Protogen Coverage

- ☑ General Universal Protogen
- ☐ Custom Protogen …………………………………..

ProtoOpen Release Note

Version	Description
1.0	Initial Release

Table of Content

Getting Start

ProtoOpen is a feature to allow Protogen owners to develop their own feature, the program must write in C Language only. The code will compile internally in your Protogen brain and store compiled code to internal storage automatically. You can choose either to run your code from UI, CommanderPort or automatically. Tools needed are any IDE of your choice or even a simple notepad and any HEX Editor for the last step.

Function API Table

Display Function API

Function-API	Description
MClearDisplayBuffer(BUFF)	Call to clear display BUFF = 0-3 display / 4 media buffer in uint8_t
MSelectWriteDisplayBuffer(BUFF)	Select buffer to write BUFF = 0-3 display / 4 media buffer in uint8_t

MWritePixel(FS, X, Y, R, G, B)	Write pixel
	FS = “LEFT” or “RIGHT” face
	X = X Axis Position of face side in uint16_t
	Y = Y Axis Position of face side in uint16_t
	R = 0-255 in uint8_t
	G = 0-255 in uint8_t
	B = 0-255 in uint8_t

MClearPixel(FS, X, Y)	Clear pixel by pixel FS = “LEFT” or “RIGHT” face X = X Axis Position of face side in uint16_t Y = Y Axis Position of face side in uint16_t
MDrawLine(FS, X, Y, L)	Draw line FS = “LEFT” or “RIGHT” face X = X Axis Position of face side in uint16_t Y = Y Axis Position of face side in uint16_t
MDrawCircle(FS, X, Y, R)	Dar circle FS = “LEFT” or “RIGHT” face X = X Axis Position of face side in uint16_t Y = Y Axis Position of face side in uint16_t
MDrawRect(FS, X1, Y1, X2, Y2)	Draw rectangular FS = “LEFT” or “RIGHT” face X1 = First X Axis Position of face side in uint16_t Y1 = Second Y Axis Position of face side in uint16_t X2 = First X Axis Position of face side in uint16_t Y2 = Second Y Axis Position of face side in uint16_t
MFillePixel(FS, X1, Y1, X2, Y2)	Fill pixel area FS = “LEFT” or “RIGHT” X1 = First X Axis Position of face side in uint16_t Y1 = Second Y Axis Position of face side in uint16_t X2 = First X Axis Position of face side in uint16_t Y2 = Second Y Axis Position of face side in uint16_t
MPushDisplayBuffer(BUFF)	Select display buffer BUFF = 0-3 display / 4 media buffer in uint8_t

Function-API	Description
MReadBuffer(BUFF, POS, &Rout, &Gout, &Bout)	Read out from buffer BUFF = 0-3 buffer n uint8_t POS = Buffer position in uint16_t &Rout = Red value return in uint8_t &Gout = Green value return in uint8_t &Bout = Blue value return in uint8_t
MWriteText(&BUF[0], FS, X, Y, SIZE)	Write text to display &BUF[0] = Text buffer in unsigned char array FS = “LEFT” or “RIGHT” face X = X Axis Position of face side in uint16_t Y = Y Axis Position of face side in uint16_t SIZE = buffer size in uint16_t

File System Function API

Function-API	Description
MFSearch(&File[0])	Check if file exists &File[0] = unsigned char array pointer to file path
MFOpen(&File[0])	Open file in file system &File[0] = unsigned char array pointer to file path
MFSize()	Check for opened file size Return file size byte in uint16_t
MFSeek(P)	Seek file P = file seek pointer position in uint32_t
MFRead(&Buff[0], len)	Read file &Buff[0] = buffer to readout in unsigned char array len = size to read in uint32_t
MFWrite(&Buff[0], len)	Write file &Buff[0] = buffer to write in unsigned char array len = size to write in uint32_t
MFmkDir(&DIR[0])	Create directory &DIR[0] = directory in unsigned char array
MFrmDir(&DIR[0])	Remove directory &DIR[0] = directory in unsigned char array

Media Function API

Function-API	Description
MPPlayAudio(&File[0])	Play audio file &File[0] = unsigned char array pointer to file path Return = media player file handle number in uint8_t
MPPlayVideo(&File[0], FS, X, Y))	Play video file &File[0] = unsigned char array pointer to file path FS = “LEFT” or “RIGHT” X = X Axis Position of face side in uint16_t Y = Y Axis Position of face side in uint16_t Return = media player file handle number in uint8_t
MPShowIMG(&File[0], FS, X, Y)	Show GIF/JPG/PNG file &File[0] = unsigned char array pointer to file path FS = “LEFT” or “RIGHT” X = X Axis Position of face side in uint16_t Y = Y Axis Position of face side in uint16_t Return = media player file handle number in uint8_t
MPStop(HANDLE)	Stop media handle HANDLE = handle number to stop in uint8_t
MPIsFinished(HANDLE)	Check if handle play finished HANDLE = handle number to check in uint8_t Return = bool

LED Strip Function API

Function-API	Description

MSColorSet(POS, R, G, B)	Set led strip color
	POS = led strip position in uint16_t
	R = 0-255 in uint8_t
	G = 0-255 in uint8_t
	B = 0-255 in uint8_t

MSClear()	Clear all led in strip

Hardware Function API

Function-API	Description
MHNoseState()	Return nose sensor state Return = bool
MHNoseRate()	Return nose sensor sensing rate Return = signed short int
MHAudChannel(CH)	Return filtered audio input CH = audio channel in uint8_t range 0-7 0 = User microphone 1 = Ambient microphone 2-7 = Auxiliary input
MHLux()	Return filtered lux level Return = lux in uint16_t
MHColorS(&R, &G, &B)	Return color detected level &R = red value in uint8_t &G = green value in uint8_t &B = blue value in uint8_t
MHGryo(&Ax, &Ay, &Az, &Gx, &Gy, &Gz)	Return gyroscope value &Ax = Accel X &Ay = Accel Y &Az = Accel Z &Gx = Gyro X &Gy = Gyro Y &Gz = Gyro Z
MHI2CWrite(ADDR, VAL)	Write I2C ADDR = I2C Address in uint8_t VAL = data to write in uint8_t
MHI2CAvailable()	Return I2C state Return = bool
MHI2CRead()	Return I2C read value Return = uint8_t
MHSerialWrite(&BUF[0], len)	Write serial &BUF[0] = buffer to write in unsigned char array len = size to write in uint32_t
MHSerialAvailable()	Return serial state Return = bool
MHSerialRead()	Read serial buffer Return = uint8_t

MHGPIOSet(PIN, TYPE)	Set GPIO
	PIN = 0-7 in uint8_t
	TYPE = bool

Control Function API

Function-API	Description
MCHoldState()	Hold function output on release/exit
MCRelease()	Exit user program and release control to OS/UI
MCSetUserProgramName(&BUF[0])	Set user program name &BUF[0] = program name in unsigned char array
MCSetProgramExecute(TYPE)	Set user program to execute from TYPE = uint8_t total number add up 1 = UI 2 = CommanderPort 4 = Other APP

Example Usage : Function API

First example, we will use Function API to read Serial then write in matrix display at position X = 10, Y = 5 on RIGHT face then exit and hold last function state. This function can be run from UI and CommanderPort.

```
void UserAPP1(void){
    // Name = TESTAPP
    unsigned char Name[] = {0x54, 0x45, 0x53, 0x54, 0x41, 0x50, 0x50, 0x00};
    unsigned char UARTBuff[100] = {0};
    unsigned int UARTSize = 0;
    unsigned char MCLock = false;
    MCSetUserProgramName(&Name[0]);      // Set this app name
    MCSetProgramExecute(3);              // Allow to run from UI and CommandPort
    while(1){
        while(MHSerialAvailable() == true){
            *(UARTBuff+UARTSize) = MHSerialRead();
            UARTSize++;
            MCLock = true;
        }
        if(MCLock == true){
            MClearDisplayBuffer(0);          // Clear matrix buffer0
            MSelectWriteDisplayBuffer(0);    // Select buffer 0 to write
            MWriteText(&UARTBuff[0], RIGHT, 10, 5, UARTSize);
            MPushDisplayBuffer(0);
            MCHoldState();
            MCRelease();
        }
    }
}
```

Remark: *MCRelease()* will break any *while* or *for* loop

Second Example, users execute from CommanderPort/UART and choose which video to play on the LEFT face then release the program to OS/UI without holding the last result.

```c
void UserAPP2(void){
    // Name = TESTAPP
    const unsigned char* RickRoll = "RICKROLL";
    const unsigned char* ThinkSMart = "THINKSMART";
    unsigned char Name[] = {0x54, 0x45, 0x53, 0x54, 0x41, 0x50, 0x50, 0x00};
    unsigned char UARTBuff[100] = {0};
    unsigned char MHandle = 0xFF;
    unsigned int UARTSize = 0;
    unsigned char MCLock = false;
    MCSetUserProgramName(&Name[0]);      // Set this app name
    MCSetProgramExecute(2);              // Allow to run from CommandPort
    while(1){
        while(MHSerialAvailable() == true){
            *(UARTBuff+UARTSize) = MHSerialRead();
            UARTSize++;
            MCLock = true;
        }
        if(MCLock == true){
            if(memcmp(&RickRoll[0], &UARTBuff[0], UARTSize) == 0){
                MHandle = MPPlayVideo("Video/RickRoll.mp4", LEFT, 0, 0);
                memset(&UARTBuff[0], 0x00, 100);
                UARTSize = 0;
            }else if(memcmp(&RickRoll[0], &UARTBuff[0], UARTSize) == 0){
                MHandle = MPPlayVideo("Video/ThinkSmartV.mp4", LEFT, 0, 0);
                memset(&UARTBuff[0], 0x00, 100);
                UARTSize = 0;
            }
            if(MHandle != 0xFF){
                if(MPIsFinished(MHandle) == true){
                    MCRelease();
                }
            }
        }
    }
}
```

Interrupt API

Coming Soon…

Interrupt API is in development, and will be released soon.

Upload Your Code & Compile

Connect to your Protogen through CommanderPort and using our software to upload ‘.C’ , our software will upload and execute compile commands automatically.

Manual Upload

Using this command to create a file

```
AT+MFOPEN=”FILEPATH”,2
```

This return file handle number, note this. then upload by this command

```
AT+MFWRITE=HANDLE,SIZE
```

Where

HANDLE = File handle number returned from earlier command
SIZE = Size of byte

Your protogen will return

```
WRITE>
```

Then you can start to put all your Hex to serial until you reach your final byte. You can get your Hex value by any Hex Editor software

Then use this command to start compile

```
AT+GCC=”FILEPATH”
```

Where

FILEPATH = is your path to your code file

There will be a URC when the compile is finished. And last, reboot to install your code.

```
AT+MREBOOT
```

IMPORTANT NOTICE AND DISCLAIMER

Contact & Support

MangMuang’s Elektronik
Thailand

Tel: -

Facebook : MangMuang’s Elektronik
Twitter : MangMuang
X : MangMuang