LOCOSYS Technology Inc.
20F.-13, No.79, Sec. 1, Xintai 5th Rd.,
Xizhi District, New Taipei City 221, Taiwan
☏ 886-2-8698-3698
886-2-8698-3699
www.locosystech.com

EPO Implement Demo Code

Vision 0.1

2023/5/26

LOCOSYS Technology Inc.
20F.-13, No.79, Sec. 1, Xintai 5th Rd.,
Xizhi District, New Taipei City 221, Taiwan
☏ 886-2-8698-3698
886-2-8698-3699
www.locosystech.com

LOCOSYS Technology Inc.
20F.-13, No.79, Sec. 1, Xintai 5th Rd.,
Xizhi District, New Taipei City 221, Taiwan
☏ 886-2-8698-3698
886-2-8698-3699
www.locosystech.com

1.0. EPO Demo

Users can reference the following Demo code for implementing EPO customization.

1.1. How to Download EPO File

User can download EPO files from http://wpepodownload.mediatek.com/ by http protocol.

It needs a vendor_ID, a project ID, a device_ID as security identity. So please submit to the E-service to apply for an EPO download account.

Final URL is:

http://wpepodownload.mediatek.com/EPO_GPS_3_1.DAT?vendor=<Vendor_ID>&project=<Project_ID>&device_id=<Device_ID>

File Name List:

EPO_GPS_3_1.DAT – GPS Only 3 Days EPO File, index 1. (For current day ~ 3 day)

http://wpepodownload.mediatek.com/EPO_GPS_3_1.DAT?vendor=<Vendor_ID>&project=<Project_ID>&device_id=<Device_ID>

EPO_GPS_3_2.DAT – GPS Only 3 Days EPO File, index 2. (For 4 day ~ 6 day)

http://wpepodownload.mediatek.com/EPO_GPS_3_2.DAT?vendor=<Vendor_ID>&project=<Project_ID>&device_id=<Device_ID>

EPO_GPS_3_3.DAT – GPS Only 3 Days EPO File, index 3. (For 7 day ~ 9 day)

http://wpepodownload.mediatek.com/EPO_GPS_3_3.DAT?vendor=<Vendor_ID>&project=<Project_ID>&device_id=<Device_ID>

EPO_GPS_3_4.DAT – GPS Only 3 Days EPO File, index 4. (For 10 day ~ 12 day)

http://wpepodownload.mediatek.com/EPO_GPS_3_4.DAT?vendor=<Vendor_ID>&project=<Project_ID>&device_id=<Device_ID>

EPO_GPS_3_5.DAT – GPS Only 3 Days EPO File, index 5. (For 13 day ~ 15 day)

http://wpepodownload.mediatek.com/EPO_GPS_3_5.DAT?vendor=<Vendor_ID>&project=<Project_ID>&device_id=<Device_ID>

EPO_GR_3_1.DAT – GPS + GLONASS 3 Days EPO File, index 1. (For current day ~ 3 day)

http://wpepodownload.mediatek.com/EPO_GR_3_1.DAT?vendor=<Vendor_ID>&project=<Project_ID>&device_id=<Device_ID>

EPO_GR_3_2.DAT – GPS + GLONASS 3 Days EPO File, index 2. (For 4 day ~ 6 day)

http://wpepodownload.mediatek.com/EPO_GR_3_2.DAT?vendor=<Vendor_ID>&project=<Project_ID>&device_id=<Device_ID>

EPO_GR_3_3.DAT – GPS + GLONASS 3 Days EPO File, index 3. (For 7 day ~ 9 day)

http://wpepodownload.mediatek.com/EPO_GR_3_3.DAT?vendor=<Vendor_ID>&project=<Project_ID>&device_id=<Device_ID>

EPO_GR_3_4.DAT – GPS + GLONASS 3 Days EPO File, index 4. (For 10 day ~ 12 day)
http://wpepodownload.mediatek.com/EPO_GR_3_4.DAT?vendor=<Vendor_ID>&project=<Project_ID>&device_id=<Device_ID>

EPO_GR_3_5.DAT – GPS + GLONASS 3 Days EPO File, index 5. (For 13 day ~ 15 day)

http://wpepodownload.mediatek.com/EPO_GR_3_5.DAT?vendor=<Vendor_ID>&project=<Project_ID>&device_id=<Device_ID>

QG_R.DAT – GPS + GLONASS QEPO File. (For Current hour + 6 hours)

http://wpepodownload.mediatek.com/QG_R.DAT?vendor=<Vendor_ID>&project=<Project_ID>&device_id=<Device_ID>

EPO_BDS_3.DAT – Beidou 3 Days EPO File. (For current day ~ 3 day)

http://wpepodownload.mediatek.com/EPO_BDS_3.DAT?vendor=<Vendor_ID>&project=<Project_ID>&device_id=<Device_ID>

EPO_GAL_3.DAT – Galileo 3 Days EPO File. (For current day ~ 3 day)

http://wpepodownload.mediatek.com/EPO_GAL_3.DAT?vendor=<Vendor_ID>&project=<Project_ID>&device_id=<Device_ID>

EPO_GAL_7.DAT – Galileo 7 Days EPO File. (For current day ~ 7 day)

http://wpepodownload.mediatek.com/EPO_GAL_7.DAT?vendor=<Vendor_ID>&project=<Project_ID>&device_id=<Device_ID>

QGPS.DAT – GPS Only QEPO File. (For Current hour + 6 hours)

http://wpepodownload.mediatek.com/QGPS.DAT?vendor=<Vendor_ID>&project=<Project_ID>&device_id=<Device_ID>

QBD2.DAT – Beidou Only QEPO File. (For Current hour + 6 hours)

http://wpepodownload.mediatek.com/QBD2.DAT?vendor=<Vendor_ID>&project=<Project_ID>&device_id=<Device_ID>

QGA.DAT – Galileo Only QEPO File. (For Current hour + 6 hours)

http://wpepodownload.mediatek.com/QGA.DAT?vendor=<Vendor_ID>&project=<Project_ID>&device_id=<Device_ID>

LOCOSYS Technology Inc.
20F.-13, No.79, Sec. 1, Xintai 5th Rd.,
Xizhi District, New Taipei City 221, Taiwan
☏ 886-2-8698-3698
886-2-8698-3699
www.locosystech.com

1.2. How to Send Host Aiding

Users can refer to mcu\project\ag33xx_evk\apps\gnss_demo\src\epo_demo to do the EPO host aiding. You can refer to the following demo to see how to proceed with the procedure.

```
void epo_demo_send_data(epo_demo_epo_data_t *data_p, int32_t data_num,
int32_t  type){

    char temp_buffer[MNL_SERVICE_MAX_COMMAND_LEN] = {0};
    uint8_t data_buffer[EPO_DEMO_RECORD_SIZE] = {0};
    int32_t i;
    int32_t sv_prn = 0;

    for(i = 0; i < data_num; i++) {
        unsigned int *epobuf = (unsigned int *)data_buffer;
        epo_demo_epo_fread(data_p, data_buffer, EPO_DEMO_RECORD_SIZE);
        sv_prn = epo_demo_get_sv_prn(type, data_buffer);

 sprintf((char *) temp_buffer,
     "471,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X",
(unsigned int)type,
                (unsigned int)sv_prn,
                epobuf[0], epobuf[1], epobuf[2], epobuf[3], epobuf[4],
epobuf[5],
                epobuf[6], epobuf[7], epobuf[8], epobuf[9], epobuf[10],
epobuf[11],
               epobuf[12], epobuf[13], epobuf[14], epobuf[15], epobuf[16],
epobuf[17]);

        gnss_app_send_command_ex(temp_buffer);
        memset(temp_buffer, 0, MNL_SERVICE_MAX_COMMAND_LEN);
#ifndef _MSC_VER
        vTaskDelay(10);
#endif
    }
}
```

LOCOSYS Technology Inc.
20F.-13, No.79, Sec. 1, Xintai 5th Rd.,
Xizhi District, New Taipei City 221, Taiwan
☏ 886-2-8698-3698
886-2-8698-3699
www.locosystech.com

1.3. How to Send Flash Aiding

Flash aiding is based on binary format so the EPO data has to be encoded in the binary format before sending to a device. You can refer to the following demo to see how to proceed with the procedure.

```
#define GNSS_BINARY_PREAMBLE1        (0x04)
#define GNSS_BINARY_PREAMBLE2        (0x24)
#define GNSS_BINARY_ENDWORD1         (0xAA)
#define GNSS_BINARY_ENDWORD2         (0x44)
#define GNSS_BINARY_PREAMBLE_SIZE    (2)
#define GNSS_BINARY_CHECKSUM_SIZE    (1)
#define GNSS_BINARY_ENDWORD_SIZE     (2)
#define GNSS_BINARY_CONTROL_SIZE     (GNSS_BINARY_PREAMBLE_SIZE + ¥
    GNSS_BINARY_CHECKSUM_SIZE + ¥
    GNSS_BINARY_ENDWORD_SIZE)
#define GNSS_BINARY_MESSAGE_ID_SIZE      (2)
#define GNSS_BINARY_PAYLOAD_LENGTH_SIZE(2)
#define GNSS_BINARY_PAYLOAD_HEADER_SIZE(GNSS_BINARY_MESSAGE_ID_SIZE + ¥
    GNSS_BINARY_PAYLOAD_LENGTH_SIZE)
#define GNSS_EPO_GR_3D_STR     "EPO_GR_3_%d.DAT"
#define GNSS_EPO_GPS_3D_STR    "EPO_GPS_3_%d.DAT"
#define GNSS_EPO_GAL_3D_STR    "EPO_GAL_3.DAT"
#define GNSS_EPO_BDS_3D_STR    "EPO_BDS_3.DAT"

#define GNSS_QEPO_GR_STR"QG_R.DAT"    #define GNSS_QEPO_GPS_STR   "QGPS.DAT"    #define
GNSS_QEPO_BD2_STR  "QBD2.DAT"    #define GNSS_QEPO_GA_STR     "QGA.DAT"

typedef enum{
    GNSS_EPO_MODE_GPS,
    GNSS_EPO_MODE_GLONASS,
    GNSS_EPO_MODE_GALILEO,
    GNSS_EPO_MODE_BEIDOU
```

```
}gnss_epo_mode_t;

typedef enum {
        GNSS_EPO_DATA_WRITE,
        GNSS_EPO_DATA_READ,
        GNSS_EPO_DATA_APPEND
}gnss_epo_data_type;

typedef enum{
        GNSS_EPO_GPS_FILE_GPS,
        GNSS_EPO_GPS_FILE_GR
}gnss_epo_gps_file_type;

typedef struct{
        FILE *file;
        gnss_epo_data_type type;
}gnss_epo_data_t;

typedef struct {
        uint16_t message_id;
        uint16_t data_size;
        uint8_t *data;
} gnss_binary_payload_t;

#define GNSS_EPO_DATA_SIZE       (72)
#define GNSS_EPO_START_END_SIZE      (1)

#define GNSS_EPO_GPS_SV   (32)
#define GNSS_EPO_GLONASS_SV   (24)
#define GNSS_EPO_GALILEO_SV     (36)
#define GNSS_EPO_BEIDOU_SV      (63)
```

```
typedef enum{
    GNSS_EPO_DATA_EPO_START      = 1200, /**< The start command before EPO data
aiding. */
    GNSS_EPO_DATA_EPO_DATA = 1201, /**< The EPO data need to be saved to the
GNSS chip. */
    GNSS_EPO_DATA_EPO_END   = 1202      /**< The end command after EPO aiding.
*/ }gnss_epo_data_id_t;

static const uint8_t g_gnss_epo_payload_gps    = 'G';    static const uint8_t g_gnss_epo_payload_glonass = 'R';
static const uint8_t g_gnss_epo_payload_galileo = 'E';    static const uint8_t g_gnss_epo_payload_beidou    =
'C';

int32_t g_gnss_epo_gps_file_type = GNSS_EPO_GPS_FILE_GPS;   /* GPS EPO file*/

uint8_t gnss_binary_calculate_binary_checksum(const gnss_binary_payload_t *const    payload)
{
    uint8_t checksum = 0;
    uint8_t *pheader = NULL;
    uint8_t *pdata = NULL;
    uint16_t i;

    if ((NULL == payload) || (payload->data == NULL)) {
        return 0;
    }
    /* The checksum is the 8-bit exclusive OR of all bytes in the payload. */


    pheader = (uint8_t*)payload;
    for (i = 0; i < GNSS_BINARY_PAYLOAD_HEADER_SIZE; i++){
        checksum ^= *pheader;
    pheader++;
```

```
        }

        pdata = (uint8_t*)payload->data;

        for (i = 0; i < payload->data_size; i++){
                checksum ^= *pdata;
        pdata++;
        }

        return checksum;
}

int16_t gnss_binary_encode_binary_packet(uint8_t *const buffer, uint16_t    max_buffer_size, const
gnss_binary_payload_t *const payload)
{
        uint8_t *pbyte;
        uint16_t required_length;

        if ((NULL == buffer) || (payload == NULL) || (payload->data == NULL)) {
                printf("binary encode: NULL buffer¥r¥n");
        return -1;
        }

        required_length = payload->data_size + GNSS_BINARY_CONTROL_SIZE +
GNSS_BINARY_PAYLOAD_HEADER_SIZE;
        if (max_buffer_size < required_length){
        printf("binary encode: wrong parameters¥r¥n");
        return -1;
        }

        memset((void *)buffer, 0 , max_buffer_size);
```

```
	buffer[0] = GNSS_BINARY_PREAMBLE1;
	buffer[1] = GNSS_BINARY_PREAMBLE2;

	pbyte = &buffer[2];
	memcpy(pbyte, payload, GNSS_BINARY_PAYLOAD_HEADER_SIZE);

	pbyte += GNSS_BINARY_PAYLOAD_HEADER_SIZE;
	memcpy(pbyte,payload->data,payload->data_size);

	pbyte += payload->data_size;

	*pbyte++ = gnss_binary_calculate_binary_checksum(payload);
	*pbyte++ = GNSS_BINARY_ENDWORD1;
	*pbyte = GNSS_BINARY_ENDWORD2;

	return required_length;
}

bool gnss_epo_fopen(gnss_epo_data_t *data_p, char* file_name, uint8_t type)
{

	if (type == GNSS_EPO_DATA_WRITE) {
		data_p->file = fopen(file_name, "wb");
	} else if (type == GNSS_EPO_DATA_READ) {
		data_p->file = fopen(file_name, "rb");
	} else {
		data_p->file = fopen(file_name, "ab");
	}
	if (data_p->file == NULL){
	return false;
```

```
        }
        data_p->type = type;
        return true;
}

int32_t gnss_epo_fread(gnss_epo_data_t *data_p, void *buff, int32_t len)
{
        int32_t ret = 0;
        ret = fread(buff, 1, len, data_p->file);
        return ret;
}

int32_t gnss_epo_fseek(gnss_epo_data_t *data_p, int32_t offset)
{
        return (int32_t)fseek(data_p->file, offset, SEEK_SET);
}

int32_t gnss_epo_fwrite(gnss_epo_data_t *data_p, void *buff, int32_t len)
{
        int32_t ret = 0;
        ret = fwrite(buff, 1, len, data_p->file);
        return ret;
}

void gnss_epo_fclose(gnss_epo_data_t *data_p)
{
        fclose(data_p->file);
}

int16_t gnss_epo_encode_binary(uint8_t *const buffer, uint16_t max_buffer_size,    uint8_t *const
temp_buffer, gnss_epo_data_id_t msg_id)
```

```
{
	gnss_binary_payload_t payload = {0};
	int16_t length = 0;
	payload.message_id = msg_id;
	if ((msg_id == GNSS_EPO_DATA_EPO_START) || (msg_id == GNSS_EPO_DATA_EPO_END)) {
		payload.data_size = GNSS_EPO_START_END_SIZE;
	} else if (msg_id == GNSS_EPO_DATA_EPO_DATA) {
	payload.data_size = GNSS_EPO_DATA_SIZE;
	} else {
	printf("gnss_epo_encode_binary, msg_id wrong¥r¥n");
	return 0;
	}
	payload.data = temp_buffer;
	length = gnss_binary_encode_binary_packet(buffer, max_buffer_size, &payload);
	return length;
}




void gnss_epo_flash_aiding(gnss_epo_mode_t type)
{
	gnss_epo_data_t data;
	uint8_t buffer[100];
	uint8_t temp_buffer[GNSS_EPO_DATA_SIZE];
	char file_name[125];
	uint8_t payload_type;
	int16_t length = 0;
	int32_t segment = 0;
	int32_t skip_length = 0;
```

```
int32_t max_sv = 0;
memset(buffer, 0, sizeof(buffer));
memset(temp_buffer, 0, sizeof(temp_buffer));
memset(file_name, 0, sizeof(file_name));
switch (type) {
case GNSS_EPO_MODE_GPS: {
if (g_gnss_epo_gps_file_type == GNSS_EPO_GPS_FILE_GR) {
        sprintf(file_name, GNSS_EPO_GR_3D_STR, 1);
skip_length = GNSS_EPO_GLONASS_SV*GNSS_EPO_DATA_SIZE;
max_sv = GNSS_EPO_GPS_SV;
} else {
sprintf(file_name, GNSS_EPO_GPS_3D_STR, 1);
}
payload_type = g_gnss_epo_payload_gps;
break;
}
case GNSS_EPO_MODE_GLONASS: {
        sprintf(file_name, GNSS_EPO_GR_3D_STR, 1);
        payload_type = g_gnss_epo_payload_glonass;
        skip_length = GNSS_EPO_GPS_SV*GNSS_EPO_DATA_SIZE;
        max_sv = GNSS_EPO_GLONASS_SV;
break;
}
case GNSS_EPO_MODE_GALILEO: {
        sprintf(file_name, GNSS_EPO_GAL_3D_STR);
        payload_type = g_gnss_epo_payload_galileo;
        break;
}
case GNSS_EPO_MODE_BEIDOU: {
        sprintf(file_name, GNSS_EPO_BDS_3D_STR);
        payload_type = g_gnss_epo_payload_beidou;
```

```
        break;
    }
    default: {
    printf("gnss_epo_flash_aiding, type wrong¥r¥n");
    return;
    }
    }

    if (gnss_epo_fopen(&data, file_name, GNSS_EPO_DATA_READ)) {
        temp_buffer[0] = payload_type;
    length = gnss_epo_encode_binary(buffer, sizeof(buffer), temp_buffer, GNSS_EPO_DATA_EPO_START);
    //gnss_bridge_put_data(buffer, length); /* aiding EPO */
    Sleep(10); /* delay 10ms*/




    if (type == GNSS_EPO_MODE_GLONASS) {
    if (fseek(data.file, skip_length, SEEK_CUR) == -1) {
        gnss_epo_fclose(&data);
    printf("gnss_epo_flash_aiding, seek fail¥r¥n");
    return;
    }
    }
    while(gnss_epo_fread(&data, temp_buffer, GNSS_EPO_DATA_SIZE)) {
        length = gnss_epo_encode_binary(buffer, sizeof(buffer), temp_buffer,
GNSS_EPO_DATA_EPO_DATA);
    //gnss_bridge_put_data(buffer, length); /* aiding EPO */
    if (((type == GNSS_EPO_MODE_GPS) && (g_gnss_epo_gps_file_type ==  GNSS_EPO_GPS_FILE_GR)) ||
(type == GNSS_EPO_MODE_GLONASS)) {
```

```
            segment++;
        if (segment >= max_sv) {
        segment = 0;
        if (fseek(data.file, skip_length, SEEK_CUR) == -1) {
        gnss_epo_fclose(&data);
        printf("gnss_epo_flash_aiding, seek fail¥r¥n");
        return;
        }
        }
        }
        Sleep(10); /* delay 10ms*/
        }
        temp_buffer[0] = payload_type;
        length = gnss_epo_encode_binary(buffer, sizeof(buffer), temp_buffer,    GNSS_EPO_DATA_EPO_END);
        //gnss_bridge_put_data(buffer, length); /* aiding EPO */
        } else {
        printf("gnss_epo_flash_aiding, no %d EPO file¥r¥n", type);
        return;
        }
        printf("gnss_epo_flash_aiding finish: %d¥r¥n", type);
        gnss_epo_fclose(&data);
}
```