Using FCM Flow in an Application

2170037-4.0

Author:	Sierra Wireless	Date:	January 25, 2012

APN Content Level	BASIC		INTERMEDIATE		ADVANCED	✓	Confidentiality	Public	✓	Private	

Hardware Compatibility	Product Line	AirPrime	Series	Q26xx	SL60xx
				WMPxx	Q26Extreme

Software Compatibility	Series	Q26 : >6.6x
		Others : ALL

1 Version

Application Notes may be updated over their lifetime. To ensure you design with the correct version, please check the application notes page in www.sierrawireless.com for latest versions.

2 Introduction

This application note describes the steps to open a FCM flow (V24 serial flow, GSM flow or GPRS flow) in an application.

This document also gives details on a sample which implements an AT custom command to read data from the flash and send them to the external application using the FCM V24 serial flow.

This Application Note (APN) is provided to Sierra Wireless distributors and clients to aid more rapid development of embedded applications using the Sierra Wireless portfolio of cellular solutions. To request a new application note, contact your regional Sierra Wireless Product Marketing Manager.

3 Glossary

Initials	Definition
DLC	Data Link Connection
FCM	Flow Control Manager
TE	Terminal Equipment
UE	User Equipment (Sierra Wireless embedded module)

4 Overview

FCM (Flow Control Manager) service to handle all FCM events and to access to the data ports provided on the product.

The Flow Control Manager APIs provides the Input-Output flows to the embedded application.

The Flow Control Manager APIs support four different I/O flows:

1. V24 serial flow (the serial link can be switched in AT mode or Data mode)
2. GSM data flow
3. GPRS data flow
4. CMUX

The embedded application can subscribe and unsubscribe to the different I/O flows.

The embedded application should subscribe to the Bluetooth virtual data ports or V24 serial flow, in order to exchange data with the external application through the serial link.

The embedded application should subscribe to the GSM CSD call data port, GPRS session data port, in order to exchange data through the GSM or GPRS data channel.

If the application does not use the FCM service, all of the embedded module ports are processed by the firmware by default.

The default behaviors are:

- When a GSM CSD call is set up, the GSM CSD data port is directly connected to the UART port where the ATD command was sent;
- When a GPRS session is set up, the GPRS data port is directly connected to the UART port where the ATD or AT+CGDATA command was sent;

- When a Bluetooth peripheral is detected & connected through an SPP based profile, a local data bridge may be set up between a Bluetooth virtual data port and the required UART port, using the AT+WLDB command.

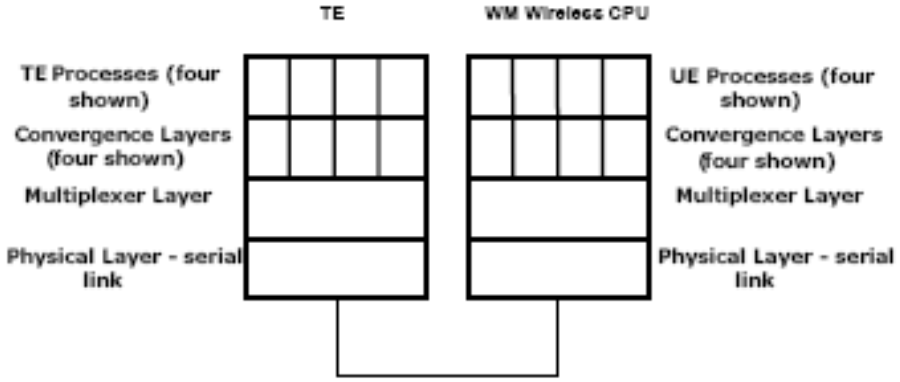
Once subscribed by an application with the FCM service, a port is no longer available to be used with the AT commands by an external application. The available ports are the ones listed in the ADL AT/FCM Ports service:

- ADL_PORT_UART_X/ADL_PORT_UART_X_VIRTUAL_BASE identifiers may be used to access the module's physicals UARTS, or logical 27.010 protocol ports;
- ADL_PORT_GSM_BASE identifier may be used to access a remote modem (connected through a GSM CSD call) data flow;
- ADL_PORT_GPRS_BASE identifier may be used to exchange IP packets with the operator network and the Internet;ADL_PORT_BLUETOOTH_VIRTUAL_BASE may be used to access a connected Bluetooth device data stream with the Serial Port Profile (SPP).

5 FCM flow over CMUX

The CMUX feature allows multiple virtual ports to be created on a physical serial link, like the multiplexing of 4 logical channels on a single UART. Multiplexing can be applied to only one UART though two UARTs are available on the Sierra Wireless embedded module. By this client application may handle up to 5 channels (4 logical multiplexed channels on an UART and 1 physical channel on the other UART). 4 DLCs are allowed to open by the Sierra Wireless embedded module when a CMUX session is enabled. These DLCs can be in the range of 1 to 4. This excludes DLC0 which is for the control channel. These ports can be used for maintaining several simultaneous sessions (SMS, Call, GPRS etc) on the same serial interface.

A virtual connection is established between TE and UE processes during a CMUX session. Communication is established using a multiplexing layer. This layer opens multiple channels over a single serial link. These channels are called data link connection (DLC). Convergence layers are used to carry the state of V24 control signals through a DLC.



When CMUX is used, the response to AT commands will be received on the virtual port where the commands were issued. Only unsolicited messages will be transmitted to all logical channels.

While executing the application from the logical ports multiplexed on the required UART port, then the port parameter used with any API should be "ADL_PORT_UART_X_VIRTUAL_BASE", and should be "OR" ed with the logical port through which you want the function to be executed. For example, if logical ports are multiplexed on UART1, then in the port parameter "ADL_PORT_UART1_VIRTUAL_BASE | 0x02", the "0x02"specifies the logical port (the DLC number) through which the function should proceed.

6 FCM functions

adl_fcm.h header file must be included when the application using flow control is developed in ADL mode.

6.1. adl_fcmSubscribe function

This function is used to subscribe to the FCM flow, open the requested flow and set the control and data handler for the flow. The subscription will be successful only when control event handler has received the event ADL_FCM_EVENT_FLOW_OPENED.

Prototype

s8 adl_fcmSubscribe (adl_fcmFlow_e Flow, adl_fcmCtrlHdlr CtrlHandler, adl_fcmDataHdlr Data Handler)

Flow

Flow determines which flow is to be opened: GSM data flow, GPRS data flow or V24 flow on UART1 or UART2. The various options are:

ADL_FCM_FLOW_GSM_DATA
ADL_FCM_FLOW_GPRS
ADL_FCM_FLOW_V24_UART1
ADL_FCM_FLOW_V24_UART2
ADL_FCM_FLOW_V24_USB

Note: For serial-link related flows (ADL_FCM_FLOW_V24_UART1 & 2), the corresponding UART has to be opened first with the "AT+WMFM" command, otherwise the subscription will fail. By default, only the UART1 is opened. The V24 serial flow is in master mode as default mode.

Note: The main difference between a subscription in master mode (ADL_FCM_FLOW_V24_UART1) and a subscription in slave mode (ADL_FCM_FLOW_V24_UART1 | ADL_FCM_FLOW_SLAVE) is that, in the first case the application can switch between the AT-mode and the Data-Mode but in the second case, the application can't switch the serial link state.

Note: ADL_FCM_FLOW_V24_MASTER corresponds to ADL_FCM_FLOW_V24_UART1 and ADL_FCM_FLOW_V24 corresponds to (ADL_FCM_FLOW_V24_UART1 | ADL_FCM_FLOW_SLAVE). These two terms are kept for compatibility with OS v2.xx.

CtrlHandler

All the events related to the FCM flow are reported in this handler; for example on successful subscription of FCM flow, the event generated is ADL_FCM_EVENT_FLOW_ OPENED.

The following table lists the flow events that could be received by the control handler:

FCM Events	Description
ADL_FCM_EVENT_FLOW_OPENNED	This event is generated when the subscribed flow is opened
ADL_FCM_EVENT_FLOW_CLOSED	This event is generated when the subscribed flow is closed.
ADL_FCM_EVENT_V24_DATA_MODE	This event is generated when the V24 flow is subscribed and Data mode is selected.
ADL_FCM_EVENT_V24_DATA_MODE_EXT	This event is generated when the Data mode switches from the external application.
ADL_FCM_EVENT_V24_AT_MODE	This event is generated when the V24 flow is subscribed and AT mode is selected
ADL_FCM_EVENT_V24_AT_MODE_EXT	This event is generated when the AT mode switches from the external application..
ADL_FCM_EVENT_RESUME	This event is related to adl_fcmSendData() API.
ADL_FCM_EVENT_MEM_RELEASE	This event is related to adl_fcmSendData() API

Data Handler

This handler receives the data block from the associated flow. Once the block is processed, the handler must return true to release the credits.

Return value

A positive or a null handle is returned on success.
ADL_RET_ERR_PARAM if one parameter has a incorrect value
ADL_RET_ERR_ALREADY_SUBSCRIBED if the flow is already subscribed in master mode.
ADL_RET_ERR_NOT_SUBSCRIBED if a slave subscription is made when the master flow is not subscribed.
ADL_FCM_RET_ERROR_GSM_GPRS_ALREADY_OPENED if a GSM or GPRS subscription is made when the other one is already subscribed.
ADL_RET_ERR_BAD_STATE if the required port is not available.
ADL_RET_ERR_SERVICE_LOCKED if the function was called from a low level interrupt handler (Applicable since OS v4.1x only).

6.2. The adl_fcmUnsubscribe function

This API unsubscribes the previously subscribed FCM service and closes the previously opened flow. The unsubscription will be effective only when the control event handler has received ADL_FCM_EVENT_FLOW_CLOSED event.

Prototype

s8 adl_fcmUnsubscribe (u8 handle)

Handle

Handle returned by the adl_fcmSubscribe function.

Return value

OK on success.
ADL_RET_ERR_UNKNOWN_HANDLE if the handle is incorrect,
ADL_RET_ERR_NOT_SUBSCRIBED if the flow is already subscribed.
ADL_RET_ERR_BAD_STATE if the serial link is not in AT mode,
ADL_RET_ERR_SERVICE_LOCKED if the function was called from a low level interrupt handler (Applicable since OS v4.1x only).

6.3. The adl_fcmSwitchV24State function

This function switches the V24 serial link state to AT mode or to Data mode. The operation will be effective only when the control event handler has received an ADL_FCM_EVENT_V24_XXX_MODE event. Only the V24 flow which has been subscribed in master mode (default mode) can use this API and switch the V24 serial link state.

Prototype

s8 adl_fcmSwitchV24State (u8 Handle, u8 V24State)

Handle

Handle returned by the adl_fcmSubscribe function.

V24State

Serial link state to switch to. Allowed values are:

ADL_FCM_V24_STATE_AT
ADL_FCM_V24_STATE_DATA

Return value

OK on success.
ADL_RET_ERR_PARAM if one parameter has a incorrect value,
ADL_RET_ERR_UNKNOWN_HDL if the provided handle is unknown
ADL_RET_ERR_BAD_HDL if the handle is not the main flow one.
ADL_RET_ERR_SERVICE_LOCKED if the function was called from a low level interrupt handler (Applicable since OS v4.1x only).

6.4. The adl_fcmSendData function

The function sends a data block on requested flow. The data block should be allocated and released by the application.

Prototype

s8 adl_fcmSendData (u8 Handle, u8 * Data, u16 DataLen)

Handle

Handle returned by the adl_fcmSubscribe function.

Data

Data block buffer to write.

The block is not released by adl_fcmSendData() API.

DataLen

Length of data to send.

Return value

OK on success.
ADL_FCM_RET_OK_WAIT_RESUME on success, but the last credit was used. The control handler will also receive a ADL_FCM_EVENT_MEM_RELEASE event when the data block memory buffer will be released.
ADL_RET_ERR_PARAM is a parameter has a incorrect value.
ADL_RET_ERR_UNKNOWN_HDL if the provided handle is unknown
ADL_RET_ERR_BAD_STATE if the flow is not ready to send data,
ADL_FCM_RET_ERR_WAIT_RESUME if the flow has no more credits to use
ADL_RET_ERR_SERVICE_LOCKED if the function was called from a low level interrupt handler (Applicable since OS v4.1x only).

If ADL_FCM_RET_XXX_WAIT_RESUME is returned, the subscriber has to wait for ADL_FCM_EVENT_RESUME event in the control handler to continue to send data.

6.5. The adl_fcmSendDataExt function

The function sends a data block on the requested flow. The data block should be allocated and filled by the application and it is released by the API.

Prototype

s8 adl_fcmSendExt (u8 Handle, adl_fcmDataBlock *DataBlock)

Handle

Handle returned by the adl_fcmSubscribe function.

Data Block

Data Block buffer to write. Using the following type:

```
Typedef struct
{
u16 Reserved1[4];
u16 DataLength;    // length of data to send
u16 Reserved2 [5];
u8   Data [1];        //data to send;
} adl_fcmDataBlock
```

This block must be dynamically allocated and filled by application.

The allocation size has to be **sizeof (adl_fcmDataBlock_t) + DataLength**, where DataLength is the value to be set in the **DataLength** field of the structure.

Return value

OK on success.
ADL_FCM_RET_OK_WAIT_RESUME on success, but the last credit was used. The control handler will also receive a ADL_FCM_EVENT_MEM_RELEASE event when the data block memory buffer will be released.
ADL_RET_ERR_PARAM is a parameter has a incorrect value.
ADL_RET_ERR_UNKNOWN_HDL if the provided handle is unknown
ADL_RET_ERR_BAD_STATE if the flow is not ready to send data,
ADL_FCM_RET_ERR_WAIT_RESUME if the flow has no more credits to use.
ADL_RET_ERR_SERVICE_LOCKED if the function was called from a low level interrupt handler (Applicable since OS v4.1x only).

If ADL_FCM_RET_XXX_WAIT_RESUME is returned, the subscriber has to wait for ADL_FCM_EVENT_RESUME event in the control handler to continue to send data.

Note: Difference between adl_fcmSendDataExt and adl_fcmSendData functions

- **Memory allocation:** When the "adl_fcmSendDataExt" API is used, the data block that will be sent, must be dynamically allocated and filled by the application before calling the API. The datablock will be sent to the subscribed

flow and will be released by the API on getting an "OK" or "ADL_FCM_RET_OK_WAIT_RESUME" return value. When the "adl_fcmSendData" API is used, the Datablock is allocated and filled inside the function.

- **Real time:** The "adl_fcmSendDataExt" API does not perform any processing on the provided data block which is sent directly to the flow. Therefore ‘‘adl_fcmSendDataExt" API has a shorter process time than "adl_fcmSendData" API.

6.6. The adl_fcmReleaseCredits function

This function releases the credits for requested flow handle. The ‘credits’ represent the number of data blocks received by the embedded application and sent through FCM layer.

When the data handler has released a credit (data handler return TRUE), it means that the embedded application has successfully processed the data block received.

The term credit is also used while sending the data block by the embedded application. When ADL_FCM_EVENT_RESUME is received, it means that the credits are released and FCM layer is ready to send more data.

Prototype

s8 adl_fcmReleaseCredits (u8 Handle, u8 NbCredits)

Handle

Handle returned by the adl_fcmSubscribe function.

NbCredits

Number of credits to be released for this flow. If this number is greater than the number of previously received data block, all credits are released.

Note: By calling the function with a number of credits set to 0xFF, the release of all credits is performed.

Return value

OK on success.
ADL_RET_ERR_UNKNOWN_HDL if the provided handle is unknown
ADL_RET_ERR_BAD_HDL if the handle is slave one.
ADL_RET_ERR_SERVICE_LOCKED if the function was called from a low level interrupt handler (Applicable since OS v4.1x only).

6.7. The adl_fcmGetStatus function

This function gets the buffer status (empty or not) for the requested flow handle in the requested way (I/O flow).

Prototype

s8 adl_fcmGetStatus (u8 Handle, adl_fcmWay_e Way)

Handle

Handle returned by the adl_fcmSubscribe function.

Way

This parameter defines the direction for which the buffer status is requested.

```
Typedef enum
{
        ADL_FCM_WAY_FROM_EMBEDDED,
        ADL_FCM_WAY_TO_ EMBEDDED
} adl_fcmWay_e
```

Return values

ADL_FCM_RET_BUFFER_EMPTY: If requested flow and way buffer is empty.
ADL_FCM_RET_BUFFER_NOT_EMPTY: If requested buffer is not empty.
ADL_FCM_RET_UNKNOWN_HDL: If provided handle is unknown.
ADL_RET_ERR_PARAM: If way parameter value is out of range.

7 Limitations of FCM

The maximum data block size that can be sent by FCM is:

- 500 bytes in remote mode
- 2 kbytes in target mode

8 Steps to use FCM for a V24 serial flow to receive and transmit data

- V24 serial Flow is subscribed using adl_fcmSubscribe () API. Each flow can be subscribed only once at a time.
- If two V24 serial flows have to be subscribed; one should be in master mode and the other in slave mode.
- On successful subscription of V24 Master FCM data flow, event ADL_FCM_EVENT_FLOW_OPENED is received
- The API adl_fcmSwitchV24State () can be used to switch between AT mode and Data mode.
  - In case of Data mode, the application can receive and send data to an external application.
  - In order to send data, use adl_fcmSendData () API. The data from the external application is received in the data handler.
  - The data that are sent with adl_fcmSendData() API should be allocated and released by the application
  - In AT mode, AT commands can be sent and received from external application.

9 Steps followed in the sample application

- In adl_main function first subscribe to CMUX flow on UART1.
- On receiving ADL_FCM_EVENT_FLOW_OPENED, subscribe for ‘‘AT+SWITCH’’ command.
- Switch the state to Data mode using AT+SWITCH=0 command.
- After sending the data, switch back to AT-mode.

10 Sample application

```
/* Global variable */
s8 Handle;
s8 fcm_Handle_Closed;
s8 retValue;
adl_tmr_t *timer_ptr;
u16 timeout_period = 10;
u8 count = 0;
#define DATA            0
#define AT   1
bool fcmCtrlH (adl_fcmEvent_e event );
bool fcmDataH ( u16 DataLen, u8 * buffer);

void FCMSwitchH ( adl_atCmdPreParser_t * paras )
{
    s8 ret;
    switch (wm_atoi(ADL_GET_PARAM ( paras, 0 )))
    {
                case DATA:
                      ret=adl_fcmSwitchV24State(Handle,  ADL_FCM_V24_STATE_DATA);
                      break;
                case AT:
                       ret=adl_fcmSwitchV24State(Handle, ADL_FCM_V24_STATE_AT);
                       break;
                default:
                       break;
     }
    TRACE((1,"Return value of adl_fcmSwitchV24State Function = %d", ret));
 }

bool fcmCtrlH (adl_fcmEvent_e event )
{
           switch(event)
```

```
        {
                case ADL_FCM_EVENT_FLOW_OPENNED:
                    TRACE (( 1, "FCM Flow Opened" ));
                    adl_atCmdSubscribe("AT+SWITCH",FCMSwitchH, ADL_CMD_TYPE_PARA | 0x11 );
                    break;
                caseADL_FCM_EVENT_V24_DATA_MODE:
                    TRACE (( 1, "Flow in Data Mode" ));
                    break;
                case ADL_FCM_EVENT_FLOW_CLOSED:
                    TRACE((1, "Flow Closed"));
                    break;
                case ADL_FCM_EVENT_V24_AT_MODE:
                    TRACE (( 1, "Flow in AT Mode" ));
                    adl_fcmReleaseCredits( Handle, 0XFF );
                    break;
                case ADL_FCM_EVENT_V24_AT_MODE_EXT:
                    TRACE (( 1, "Flow in AT Mode" ));
                    adl_fcmReleaseCredits( Handle, 0XFF );
                    break;
         }
         return TRUE;
}

bool fcmDataH ( u16 datalen, u8 * databuf)
{
        TRACE((1,"Data handler = %d", datalen ));
        return FALSE;
}

void adl_main ( adl_InitType_e InitType )
{
        TRACE (( 1, "Embedded Application : Main" ));
        Handle=adl_fcmSubscribe(ADL_PORT_UART1_VIRTUAL_BASE | 0x02 ,fcmCtrlH, fcmDataH);
        TRACE (( 1, "FCM subscription handle-->%d",Handle ));
}
```

11 Software Compatibility Matrix

List all current software configurations and compatibility with this application note.

FW	Framework/SDK	Library
6.6x	Open AT® SDK v4.2x	N/A
R6.5	Open AT Framework v1.x	N/A
R7.x	Open AT Framework v2.x	N/A

12 Reference Documents

	Filename	Comment
[1]	ADL User Guide for Open AT Framework OS	A section in this document describes FCM API's as well as its configurations.

13 Support

For direct clients: contact your Sierra Wireless FAE

For distributor clients: contact your distributor FAE

For distributors: contact your Sierra Wireless FAE

14 Document History

Version	Date	History
001	June 23, 2005	Creation
002	May 07, 2011	Update
003	March 03, 2011	CMUX update + new SWI template
004	January 25, 2012	New reference: 2170037 Old reference: WM_ASW_OAT_APN_024

15 Legal Notice