Link Street®
88E6341/88E6141

Integrated Micro Processor (IMP) Library

Not Approved by Document Control.
For Review Only.

Doc. No. MV-S800938-00 Rev. –
October 14, 2016, Draft
CONFIDENTIAL
Document Classification: Restricted Information

Marvell. Moving Forward Faster

Document Conventions

	**Note:** Provides related information or information of special importance.
 Caution	**Caution:** Indicates potential damage to hardware or software, or loss of data.
	**Warning:** Indicates a risk of personal injury.

Document Status

Doc Status: Draft	Technical Publication: 1.0.0

For more information, visit our website at: http://www.marvell.com

Table of Contents

1 Introduction

This document describes the IMP Lib of Marvell SOHO Switch Products, Application Programming Interface (API) definition and driver usage. Currently the driver supports Peridot and Topaz.

Hardware

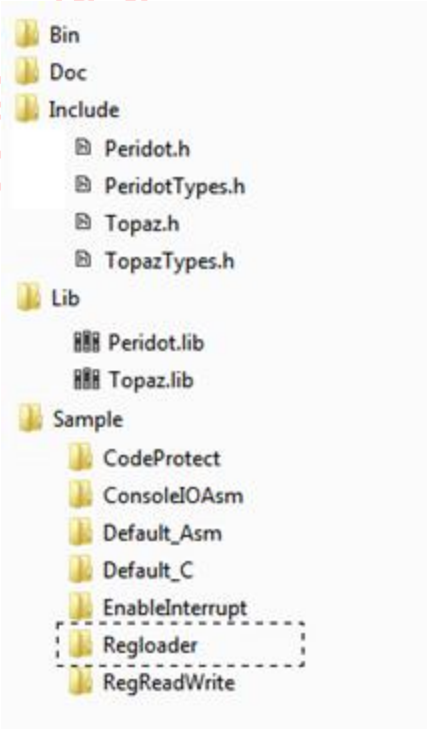
The Peridot lib just support Peridot series SOHO switch product. Please refer to switch products datasheet for more information.

Software

The lib build by SDCC 3.4.0 compiler. Reversion number is 1.0.

The lib support Module: ATU, VTU, IRL, TCAM, Common Register Access, IO, Reset

The lib need unzip to the same level directory with IMPGUI (reversion is 1.10.0 or later) as below. Only the Regloader sample use the lib files (Peridot.lib etc.)



2 API List

2.1 FDB/ATU (Filtering Database/Address Translation Unit)

2.1.1 atuAddEntry

DESCRIPTION

This routine creates a new entry in the MAC address table.

SYNOPSIS

```
char atuAddEntry
(
        ATU_ENTRY *atuEntry
)
```

ARGUMENTS

INPUTS

atuEntry - atu entry to insert

OUTPUTS

None.

RETURNS

0 - On success

2.1.2 atuFlush

DESCRIPTION

This routine flush all address from the MAC address table.

SYNOPSIS

```
char atuFlush()
```

ARGUMENTS

INPUTS

None.

OUTPUTS

None.

RETURNS

0 - On success

2.1.3 atuGetEntryNext

DESCRIPTION

This routine get next ATU entry base on user input.

SYNOPSIS

```
char atuGetEntryNext
(
        ATU_ENTRY *atuEntry
)
```

ARGUMENTS

INPUTS

atuEntry - including FID, PortVec and ATU MAC Address Database Number

OUTPUTS

atuEntry - next atu entry.

RETURNS

0 - On success

2.2 VLAN Translation Unit (VTU/802.1Q)

2.2.1 vtuAddEntry

DESCRIPTION

This routine creates the new entry in VTU table based on user input.

SYNOPSIS

```
char vtuAddEntry
(
        VTU_ENTRY  *vtuEntry
)
```

ARGUMENTS

INPUTS

vtuEntry - vtu entry to insert to the VTU table

OUTPUTS

None.

RETURNS

0 - On success

2.2.2 vtuFlush

DESCRIPTION

This routine deletes all VTU entry

SYNOPSIS

char vtuFlush()

ARGUMENTS

INPUTS

None.

OUTPUTS

None.

RETURNS

0 - On success

2.2.3 vtuGetEntryNext

DESCRIPTION

This routine find the next entry in VTU table based on user input.

SYNOPSIS

```
char msdVlanEntryDelete
(
        VTU_ENTRY  *vtuEntry
)
```

ARGUMENTS

INPUTS

vtuEntry - including vlan id

OUTPUTS

vtuEntry - next VTU entry

RETURNS

0 - On success

2.3 Ternary Content Addressable Memory (TCAM)

2.3.1 TCAMOP

DESCRIPTION

This routine cover some TCAM operation.

SYNOPSIS

```
char TCAMOP
(
        unsigned char index,
        char page,
        char op
```

)

ARGUMENTS

INPUTS

Index - index of TCAM entry

page - page of TCAM entry

op - TCAM Operation

OUTPUTS

None.

RETURNS

0 - On success

2.3.2 tcamLoadEntry

DESCRIPTION

This routine loads a TCAM entry.

The loading sequence of TCAM entry is critical. Each TCAM entry is made up of 3 pages of data. All 3 pages need to be loaded in a particular order for the TCAM to operate correctly while frames are flowing through the switch. If the entry is currently valid, it must first be flushed. Then page 2 needs to be loaded first, followed by page 1 and then finally page 0. Each page load requires its own write TCAMOp with these TCAM page bits set accordingly.

SYNOPSIS

```
char tcamLoadEntry
(
    unsigned char tcamPointer,
    TCAM_DATA  *tcamData
)
```

ARGUMENTS

INPUTS

tcamPointer - pointer to the desired entry of TCAM

tcamData - tcam entry Data

OUTPUTS

None.

RETURNS

0 - On success

2.3.3 TCAMFlushAll

DESCRIPTION

This routine is to flush all entries. A Flush All command will initialize TCAM Pages 0 and 1, offsets 0x02 to 0x1B to 0x0000, and TCAM Page 2 offset 0x02 to 0x1B to 0x0000 for all TCAM entries with the exception that TCAM Page 0 offset 0x02 will be initialized to 0x00FF.

SYNOPSIS

char TCAMFlushAll()

ARGUMENTS

INPUTS

None.

OUTPUTS

None.

RETURNS

0 - On success

2.3.4 tcamReadTCAMData

DESCRIPTION

This routine reads the global 3 offsets 0x02 to 0x1B registers with the data found in the TCAM entry and its TCAM page pointed to by the TCAM entry and TCAM page bits of this register (bits 6:0 and 11:10 respectively.

SYNOPSIS

```
char tcamReadTCAMData
(
        unsigned char tcamPointer,
        TCAM_DATA  *tcamData
)
```

ARGUMENTS

INPUTS

tcamPointer - pointer to the desired entry of TCAM

OUTPUTS

tcamData - tcam entry Data

RETURNS

0 - On success

2.3.5 tcamGetNextTCAMData

DESCRIPTION

This routine finds the next higher TCAM Entry number that is valid (i.e., any entry whose Page 0 offset 0x02 is not equal to 0x00FF). The TCAM Entry register (bits 6:0) is used as the TCAM entry to start from. To find the lowest number TCAM Entry that is valid, start the Get Next operation with TCAM Entry set to 0xFF.

SYNOPSIS

```
char tcamGetNextTCAMData
(
```

```
                unsigned char *tcamPointer,
                TCAM_DATA  *tcamData
        )
```

ARGUMENTS

INPUTS

tcamPointer - pointer to the desired entry of TCAM

OUTPUTS

tcamPointer - pointer to the desired entry of TCAM

tcamData - tcam entry Data

RETURNS

0 - On success

2.3.6 tcamLoadEgrEntry

DESCRIPTION

This routine loads a TCAM egress entry.

SYNOPSIS

```
char tcamLoadEgrEntry
(
        unsigned short tcamPointer,
        Egr_TCAM_DATA *data
)
```

ARGUMENTS

INPUTS

tcamPointer - pointer to the desired entry of TCAM

OUTPUTS

data - the entry parameters.

RETURNS

0 - On success

2.4 Ingress Rate Limiter(IRL)

2.4.1 irlInitialize

DESCRIPTION

This routine initializes all PIRL Resources for all ports.

SYNOPSIS

```
char irlInitialize()
```

ARGUMENTS

INPUTS

None.

OUTPUTS

None.

RETURNS

0 - On success

2.4.2 irlWrite

DESCRIPTION

This routine write the irl entry based on user input.

SYNOPSIS

```
char irlWrite
(
IRL_Entry *irlEntry
)
```

ARGUMENTS

INPUTS

irlEntry - target logical port

OUTPUTS

None.

RETURNS

0 - On success

2.4.3 irlRead

DESCRIPTION

This routine read Resource bucket parameter from the given resource of port.

SYNOPSIS

```
char msdIRLResourceRead
(
    IRL_Entry *irlEntry
)
```

ARGUMENTS

INPUTS

irlEntry - G2 ingress rate command

OUTPUTS

irlEntry - IRL Entry data

RETURNS

0 - On success

2.5 Common Register Access

2.5.1 ReadReg

DESCRIPTION

This function directly reads a switch's register.

SYNOPSIS

```
unsigned short ReadReg
(
        unsigned char dev,
        unsigned char reg
)
```

ARGUMENTS

INPUTS

dev	- device SMI address
reg	- The register's address

OUTPUTS

None.

RETURNS

data	- The data to be read

2.5.2 WriteReg

DESCRIPTION

This function directly writes to a switch's register.

SYNOPSIS

```
void WriteReg
(
        unsigned char   dev,
        unsigned char   reg,
        unsigned short  data
)
```

ARGUMENTS

INPUTS

dev	- device SMI address.
reg	- The register's address.
data	- The read Register’s data

OUTPUTS

None.

RETURNS

None.

2.5.3 PortRead

DESCRIPTION

This function directly reads a switch's Port register.

SYNOPSIS

```
unsigned short PortRead
(
        unsigned char port,
        unsigned char reg
)
```

ARGUMENTS

INPUTS

port	- device logical number
reg	- The register's address

OUTPUTS

None.

RETURNS

data - The data to be read

2.5.4 PortWrite

DESCRIPTION

This function directly writes to a switch's Port register.

SYNOPSIS

```
void PortWrite
(
        unsigned char   port,
        unsigned char   reg,
        unsigned char   data
)
```

ARGUMENTS

INPUTS

port	- device logical number.
reg	- The register's address.
data	- The read Register’s data

OUTPUTS

None.

RETURNS

None .

2.5.5 G1Read

DESCRIPTION

This function directly reads register of G1.

SYNOPSIS

```
unsigned short G1Read
(
        unsigned char reg
)
```

ARGUMENTS

INPUTS

reg - The register's address

OUTPUTS

None.

RETURNS

data - The data to be read

2.5.6 G1Write

DESCRIPTION

This function directly writes to register of G1.

SYNOPSIS

```
void G1Write
(
        unsigned char   reg,
        unsigned char   data
)
```

ARGUMENTS

INPUTS

reg - The register's address.

data - The read Register’s data

OUTPUTS

None.

RETURNS

None .

2.5.7 G2Read

DESCRIPTION

This function directly reads register of G2.

SYNOPSIS

```
unsigned short G2Read
(
        unsigned char reg
)
```

ARGUMENTS

INPUTS

reg - The register's address

OUTPUTS

None.

RETURNS

data - The data to be read

2.5.8 G2Write

DESCRIPTION

This function directly writes to register of G2.

SYNOPSIS

```
void G2Write
(
        unsigned char   reg,
        unsigned char   data
)
```

ARGUMENTS

INPUTS

reg - The register's address.

data - The read Register’s data

OUTPUTS

None.

RETURNS

None .

2.5.9 C45RegWrite

DESCRIPTION

This function write PHY register base on C45.

SYNOPSIS

```
void C45RegWrite
(
    unsigned char PhyAddr,
    unsigned char DevAddr,
    unsigned short Reg,
    unsigned short Data,
    unsigned char IsExternal
)
```

ARGUMENTS

INPUTS

PhyAddr - phy address
DevAddr - device register
Reg - The register's address
Data - Data to write
IsExternal - Is External

OUTPUTS

None.

RETURNS

None.

2.5.10 C45RegRead

DESCRIPTION

This function read PHY register base on C45.

SYNOPSIS

```
unsigned short C45RegRead
(
    unsigned char PhyAddr,
    unsigned char DevAddr,
    unsigned short Reg,
    unsigned char IsExternal
)
```

ARGUMENTS

INPUTS

PhyAddr - phy address
DevAddr - device register
Reg - The register's address

IsExternal - Is External

RETURNS

data – register to read

2.5.11 C22RegWrite

DESCRIPTION

This function writes PHY register base on C22.

SYNOPSIS

```
void C22RegWrite
(
    unsigned char PhyAddr,
    unsigned char Reg,
    unsigned short Data,
    unsigned char IsExternal
)
```

ARGUMENTS

INPUTS

PhyAddr - the PHY address
Reg - The register address
data - data to be written
IsExternal - Is External

OUTPUTS

None.

RETURNS

None.

2.5.12 C22RegRead

DESCRIPTION

This function reads PHY register base on C22.

SYNOPSIS

```
unsigned short C22RegRead
(
    unsigned char PhyAddr,
    unsigned char Reg,
    unsigned char IsExternal
```

)

ARGUMENTS

INPUTS

PhyAddr - the PHY address to be read

Reg - The register address to read

IsExternal - Is External

OUTPUTS

None.

RETURNS

data - the read register's value

2.5.13 ExtC22Read

DESCRIPTION

This function reads External PHY register base on C22.

SYNOPSIS

unsigned short ExtC22Read

(

unsigned char PhyAddr,

unsigned char Reg,

)

ARGUMENTS

INPUTS

PhyAddr - the PHY address to be read

Reg - The register address to read

OUTPUTS

None.

RETURNS

data - the read register's value

2.5.14 ExtC22Write

DESCRIPTION

This function writes External PHY register base on C22.

SYNOPSIS

```
void ExtC22Write
(
        unsigned char PhyAddr,
        unsigned char Reg,
        unsigned short Data,
)
```

ARGUMENTS

INPUTS

PhyAddr - the PHY address
Reg - The register address
data - data to be written

OUTPUTS

None.

RETURNS

None.

2.5.15 IntC22Read

DESCRIPTION

This function reads internal PHY register base on C22.

SYNOPSIS

```
unsigned short IntC22Read
(
        unsigned char PhyAddr,
        unsigned char Reg,
)
```

ARGUMENTS

INPUTS

PhyAddr - the PHY address to be read
Reg - The register address to read

OUTPUTS

None.

RETURNS

data - the read register's value

2.5.16 IntC22Write

DESCRIPTION

This function writes Internal PHY register base on C22.

SYNOPSIS

```
void IntC22Write
(
        unsigned char PhyAddr,
        unsigned char Reg,
        unsigned short Data,
)
```

ARGUMENTS

INPUTS

PhyAddr	- the PHY address
Reg	- The register address
data	- data to be written

OUTPUTS

None.

RETURNS

None.

2.6 IO

2.6.1 FastCopy

DESCRIPTION

This function is memory copy by DMA.

SYNOPSIS

```
void * FastCopy
(
        void *dest,
        void *src,
        unsigned int count,
        unsigned char type
)
```

ARGUMENTS

INPUTS

Dest - the destination address
src - the source address
count - the count to copy
type - RAM to RAM or EEPROM to RAM

OUTPUTS

None.

RETURNS

None.

2.6.2 memcpy_fast

DESCRIPTION

This function is the fast copy RAM to RAM.

SYNOPSIS

```
void * memcpy_fast
(
    void *dest,
    void *src,
    unsigned int count,
)
```

ARGUMENTS

INPUTS

Dest - the destination address
src - the source address
count - the count to copy

OUTPUTS

None.

RETURNS

None.

2.6.3 eeprom2ram

DESCRIPTION

This function is the fast copy EEPROM to RAM.

SYNOPSIS

```
void * eeprom2ram
(
```

```
                    void *dest,
                    void *src,
                    unsigned int count,
                )
```

ARGUMENTS

INPUTS

Dest - the destination address

src - the source address

count - the count to copy

OUTPUTS

None.

RETURNS

None.

2.7 Reset

2.7.1 ResetSwitch

DESCRIPTION

This function is to reset Switch register.

SYNOPSIS

```
void  ResetSwitch()
```

ARGUMENTS

INPUTS

None.

OUTPUTS

None.

RETURNS

None.

3 Enum and Structure

3.1 FDB/ATU (Filtering Database/Address Translation Unit)

3.1.1 ATU_ENTRY

SYNTAX

```
typedef struct
{
    unsigned char MAC_FPri :3;
    const unsigned char Res0: 1;
    const unsigned char ATUFullViolation :1;
    const unsigned char MissViolation :1;
    const unsigned char MemberViolation :1;
    const unsigned char AgeOutViolation :1;
    unsigned char MAC_QPri :3;
    const unsigned char Res1 :1;
    unsigned char ATUOp :3;
    unsigned char ATUBusy :1;

    unsigned char EntryState_SPID :4;
    const unsigned char Res2 :4;
    const unsigned char Res3: 7;
    unsigned char LAG :1;

    unsigned char ATUByte1;
    unsigned char ATUByte0;
    unsigned char ATUByte3;
    unsigned char ATUByte2;
    unsigned char ATUByte5;
    unsigned char ATUByte4;
    unsigned short FID:12;
    unsigned short PortVec:11;
} ATU_ENTRY;
```

3.2 VLAN Translation Unit (VTU/802.1Q)

3.2.1 VTU_ENTRY

SYNTAX

```
typedef struct
{
    const unsigned char SPID:5;
    const unsigned char MissViolation :1;
    const unsigned char MemberViolation :1;
    const unsigned char Res2 :1;
    const unsigned char Res3 :2;
    const unsigned char VTUMode :2;
    unsigned char VTUOp :3;
    unsigned char VTUBusy:1;

    unsigned short VID :12;
    unsigned char Valid :1;
    unsigned char Page :1;
    const unsigned char Res03 :2;

    unsigned char MTagP0:2;
    unsigned char MTagP1:2;
    unsigned char MTagP2:2;
    unsigned char MTagP3:2;
    unsigned char MTagP4:2;
    unsigned char MTagP5:2;
    unsigned char MTagP6:2;
    unsigned char MTagP7:2;


    unsigned char MTagP8:2;
    unsigned char MTagP9:2;
    unsigned char MTagP10:2;
    const unsigned char Res13:2;
    unsigned char VIDFPri :3;
    unsigned char VIDFPriOverride :1;
    unsigned char VIDQPri :3;
    unsigned char VIDQPriOverride :1;

    unsigned char FID:12;
    unsigned char VidPolicy:1;
    const unsigned char Res21 :3;
    unsigned char SID :6;
    const unsigned char Res22 :2;
    const unsigned char Res23 :4;
    unsigned char FilterMC :1;
    unsigned char FilterBC :1;
    unsigned char FilterUC :1;
    unsigned char DontLearn :1;
} VTU_ENTRY;
```

3.3 Ternary Content Addressable Memory (TCAM)

3.3.1 TCAM_DATA

SYNTAX

```
typedef struct
{
    PAGE0 page0;
    PAGE1 page1;
    PAGE2 page2;
} TCAM_DATA;
```

3.3.2 PAGE0

SYNTAX

```
typedef struct
{
    unsigned char SPV8_10:3;
    unsigned char res4:3;
    unsigned char FrameType:2;
    unsigned char MaskSPV8_10:3;
    unsigned char res5:3;
    unsigned char MaskType :2;

    unsigned char SPV0_7;
    unsigned char MaskSPV0_7;

    unsigned char PVIDH :4;
    unsigned char PPPRI :4;
    unsigned char MaskPVIDH :4;
    unsigned char MaskPPPRI :4;

    unsigned char PVIDL;
    unsigned char MaskPVIDL;

    TCAMFRAMEOCTET DA[6];
    TCAMFRAMEOCTET SA[6];
    TCAMFRAMEOCTET VLAN[4];
    TCAMFRAMEOCTET TYPE[2];
    TCAMFRAMEOCTET DATA[4];
```

```
} PAGE0;
```

3.3.3 PAGE1

SYNTAX

```
typedef struct
{
    TCAMFRAMEOCTET DATA[26];
} PAGE1;
```

3.3.4 PAGE2

SYNTAX

```
typedef struct
{
    unsigned short VID :12;
    unsigned short VIDOverride :1;
    unsigned short IC :1;
    unsigned short Int :1;
    unsigned short C :1;
    unsigned char FPRI:3;
    unsigned char FPRIOverride :1;
    unsigned char QPRI :3;
    unsigned char QPRIOverride :1;
    unsigned char NextIDFlowID;
    unsigned char DPV:10;
    unsigned char SF :1;
    unsigned char res3: 4;
    unsigned short res4;
    unsigned char EgrActionPoint :6;
    unsigned char res5:2;
    unsigned char UnknownFiltering :2;
    unsigned char VTUPage :1;
    unsigned char VTUPageOverride :1;
    unsigned char ColorMode:2;
    unsigned char DPVMode :2;
    unsigned char DSCP :6;
    unsigned char DSCPOverride :1;
    unsigned char res6 :1;
    unsigned short res7 :4;
    unsigned short LoadBlance :3;
    unsigned short LoadBlanceOverride :1;
    unsigned short FAction : 15;
    unsigned short FActionOverride :1;
} PAGE2;
```

3.3.5 Egr_TCAM_DATA

SYNTAX

```
typedef struct
{
    unsigned short EgrPort : 5;
    unsigned char res2;

    unsigned short SAMode: 3;
    unsigned short res3: 1;
    unsigned short DAMode: 2;
    unsigned short res4: 2;
    unsigned short TagMode:2;
    unsigned short TagOverride:1;
    unsigned short res5: 1;
    unsigned short FrameMode: 2;
    unsigned short FrameOverride: 1;

    unsigned short EgrVID: 12;
    unsigned short VIDMode: 2;
    unsigned short VIDOverride:1;

    unsigned short EgrFPRI:3;
    unsigned short EgrCFI:1;
    unsigned short FPRIMode :2;
    unsigned short FPRIOverride:1;
    unsigned short EgrDSCP:6;
    unsigned short DSCPMode:2;

} Egr_TCAM_DATA;
```

3.3.6 TCAMFRAMEOCTET

```
typedef struct
{
    unsigned char Data;
    unsigned char Mask;
} TCAMFRAMEOCTET;
```

3.4 Ingress Rate Limiter (IRL)

3.4.1 IRL_Entry

SYNTAX

```
typedef struct
{
    unsigned char IRLReg :4;
    const unsigned char Res7 :1;
    unsigned char IRLRes :3;
    unsigned char IRLPort: 5;
    unsigned char IRLOp :2;
    unsigned char IRLBusy :1;
    unsigned short UnknownUnicastMask :1;
    unsigned short UnknownMulticastMask :1;
```

```
    unsigned short BroadcastMask :1;
    unsigned short MulticastMask :1;
    unsigned short UnicastMask :1;
    unsigned short MGMTFrames :1;
    const unsigned short Res0 :1;
    unsigned short ARP :1;
    unsigned short FLow0_TCPData :1;
    unsigned short FLow1_TCPCtrl :1;
    unsigned short FLow2_UDP :1;
    unsigned short FLow3_NONTCPUDP :1;
    unsigned short IMS :1;
    unsigned short PolicyMirror :1;
    unsigned short PolicyTrap :1;
    unsigned char PriSelect;
    unsigned short FPri :1;
    unsigned short PriAndPT :1;
    unsigned short AcctForAll :1;
    unsigned short AcctForQCong :1;
    unsigned short AcctForGrnOvflow :1;
    unsigned short ColorAware :1;
    unsigned short SMode :1;    unsigned short BktInc : 13;
    unsigned short TCAMFlows :1;
    unsigned short  CountMode :2;
    unsigned short BktRateFactorGrn;
    unsigned long CBSLimit;
    unsigned short BktRateFactorYel;
    unsigned long EBSLimit;
    unsigned char DaAvbNrlEn :1;
    unsigned char SaAvbNrlEn :1;
    unsigned char MGMTNrlEn :1;
    unsigned char FCAction :1;
    unsigned char FCMode :1;
    unsigned char FCPri :3;
}IRL_Entry;
```

A Acronyms and Abbreviations

ATU	Address Translation Unit
VTU	VLAN Table Unit
IRL	Ingress Rate Limit
TCAM	Ternary Content Addressable Memory
C22	IEEE802.3 Clause 22
C45	IEEE802.3 Clause 45

Marvell Semiconductor, Inc.
5488 Marvell Lane
Santa Clara, CA 95054, USA

Tel: 1.408.222.2500
Fax: 1.408.988.8279

http://www.marvell.com

Marvell. Moving Forward Faster