

9-1996

## Bus Stop Based Schedule Database for the Transit Time Internet

Jun Qui

*Portland State University*

Let us know how access to this document benefits you.

Follow this and additional works at: [http://pdxscholar.library.pdx.edu/cus\\_pubs](http://pdxscholar.library.pdx.edu/cus_pubs)



Part of the [Transportation Commons](#), and the [Urban Studies and Planning Commons](#)

---

### Citation Details

Qui, Jun, "Bus Stop Based Schedule Database for the Transit Time Internet" (1996). *Center for Urban Studies Publications and Reports*. Paper 95.

[http://pdxscholar.library.pdx.edu/cus\\_pubs/95](http://pdxscholar.library.pdx.edu/cus_pubs/95)

This Report is brought to you for free and open access. It has been accepted for inclusion in Center for Urban Studies Publications and Reports by an authorized administrator of PDXScholar. For more information, please contact [pdxscholar@pdx.edu](mailto:pdxscholar@pdx.edu).

PR 100  
September 1996

# Bus Stop Based Schedule Database for the Transit Time Internet Access System

by Jun Qiu

PR100  
September 1996

This project was funded by Tri-Met and Transportation Northwest (TransNow) Regional Center under the sponsorship of the Department of Transportation UTC Grant Program. The U.S. Government and Tri-Met assume no liability for the contents or use thereof. The contents do not necessarily reflect the views or policies of TransNow, the U.S. Department of Transportation, or Tri-Met.

Center for Urban Studies  
School of Urban and Public Affairs  
Portland State University  
Portland, OR 97207-0751  
(503) 725-4020  
(503) 725-5199 FAX  
<http://www.upa.pdx.edu/CUS/>

**Contents:**

1. Introduction	1
2. Creating database in ARC/INFO	1
3. Description of the computing process	4
4. Conclusion	7
5. Appendix	11

## Introduction

Advanced Public Transportation Systems (APTS) are a subset of Intelligent Transportation Systems (ITS). Bus Dispatch Systems (BDS) are important components of APTS. Based on the Automatic Vehicle Location (AVL) technology, the BDS generates exception reports for buses not adhering to their schedule. A Transit Time Internet Access System (TTIA) is under development. It currently reports bus schedule times and bus arrival times for time points (abbreviate as TP) along routes. The objective of a subsequent version of the TTIA is to provide user with schedule time at the bus stop level. This is a report on progress to estimate stop level schedules for the use in TTIA.

The Tri-County Metropolitan Transportation District of Oregon, Tri-Met, maintains a large database system. There are more than one hundred routes, each of which is composed of a list of the stops. There are several patterns to each route. Each pattern (path) has a different sequence of stops. In one day, there are many trips which pass by each of the stops. Also there are differences among the schedules for weekday, Saturday, and holiday. To determine schedule time at the stop level during a particular time frame is a complex problem.

The Center of Urban Studies, Portland State University, is currently involved in a research ITS project, called Transit Time Internet Access (TTIA). The tasks described this paper is one part of the project. In this paper, the research is based on the existing database system of Tri-Met. The work environment is the GIS software system, ARC/INFO 7.04. The DBMS is INFO, which is a module of ARC/INFO. The program is coded in ARC Macro Language, (AML). The AML is macro language, though it does not support SQL well. The scope of this paper is limited to the task of interpolation from the schedule time of time points to the schedule time of each bus stop.

### Creating database in ARC/INFO

The database is created in INFO from data imported from the TRI-MET's. The database structure in INFO keeps the same structure as that in Tri-Met database in DB2, except for some adjustment of the sequence of the items. To interpolate the time to each stop, three tables are essential. These tables and their structures are listed as following:

#### 1. Bus stop table

COLUMN	ITEM NAME	WIDTH	OUTPUT	TYPE	N.DEC
1	VERSION_KEY	6	6	I	
7	SERVICE_TYPE *	1	1	C	
8	RTE	3	3	C	
11	DIR	1	1	C	
12	PATH_NUMBER	6	6	I	
18	LOC_ID	11	11	I	
29	STOP_DISTANCE	16	16	N	10

\* SERVICE\_TYPE: Weekday, Saturday, or Sunday and Holiday . No data is in this item at this point. It is one of the tasks to fill it.

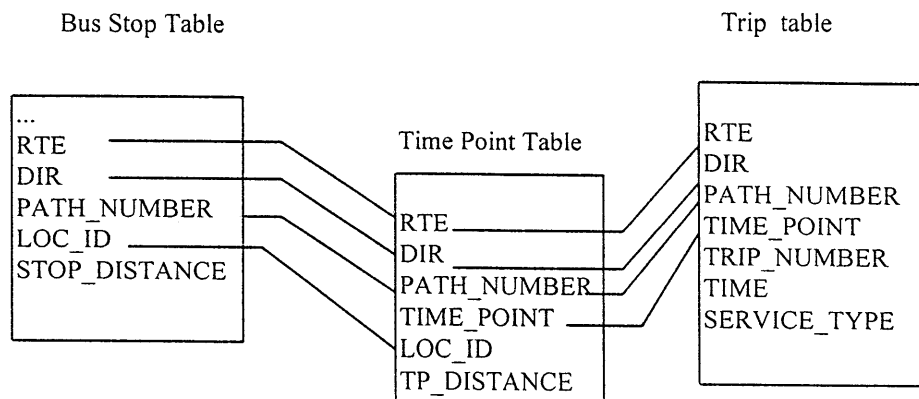
### 2. Time point table

COLUMN	ITEM NAME	WIDTH	OUTPUT	TYPE	N.DEC
1	RTE	3	3	C	
4	DIR	1	1	C	
5	PATH_NUMBER	6	6	I	
11	TIME_POINT	6	6	I	
17	LOC_ID	11	11	I	
28	TP_DISTANCE	16	16	N	10

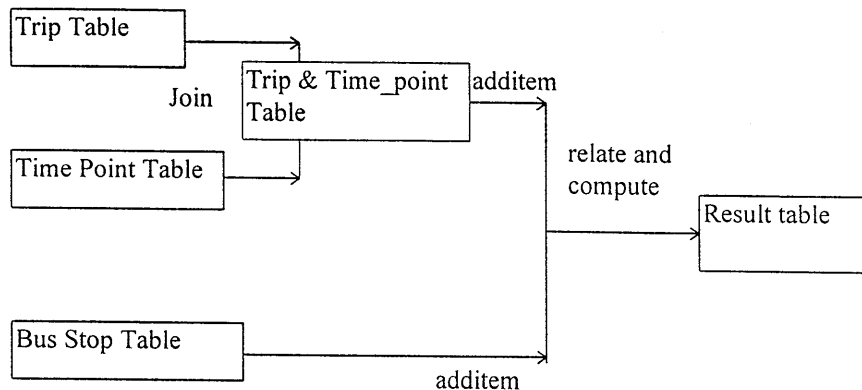
### 3. Trip table

COLUMN	ITEM NAME	WIDTH	OUTPUT	TYPE	N.DEC
1	RTE	3	3	C	
4	DIR	1	1	C	
5	PATH_NUMBER	6	6	I	
11	TIME_POINT	6	6	I	
17	TRIP_NUMBER	6	6	I	
23	TIME	11	11	I	
34	SERVICE_TYPE	1	1	C	

The database relation is created by several common items in each database tables. Their relation is described as following:



Two steps are necessary to update the database for AML programming. 1. Join the time point table to stop table. 2. Add items to the new joint table.



### Join stop table and time point table

The item TIME\_POINT is the common item to the Trip table and the Time\_point table. However the relation must be based on the same route, direction and path\_number. In this research, these two tables are joined by concatenating several items to make a new item: "rte-dir-path\_number-time\_point". Using this new item as a key warranted the joining relation. After joining the Time\_point table to Trip table, the new Trip & Time\_point table will have both time point information and trip information. This new table will be used in the following steps. And the primitive Time\_point table is no longer used.

### Add items

In ARC, the command "additem" is used to add the following items to the Trip & Time\_points table:

- DELTA\_DIS (which = tp distance - up tp distance)
- DELTA\_TIME (which = tp time - up tp distance)
- TRIP\_INCRE (Which is unique to each trip)

So far the table of Trip & Time\_point has both the trip and time, as well as the time point information. The following is its structure:

COLUMN	ITEM NAME	WIDTH	OUTPUT	TYPE	N.DEC
1	RTE	3	3	C	
4	DIR	1	1	C	
5	PATH_NUMBER	6	6	I	
11	TIME_POINT	6	6	I	
17	TRIP_NUMBER	6	6	I	
23	TIME	11	11	I	
34	SERVICE_TYPE	1	1	C	
45	LOC_ID	11	11	I	
61	TP_DISTANCE	16	16	N	10
77	DELTA_DISTANCE	16	16	N	10
88	DELTA_TIME	11	11	I	
94	TRIP_INCRE	6	6	I	

The following was added to the stop table

- UPDOWN\_TIME (to each stop, the time between its up\_tp and down\_tp)
- UPDOWN\_DIS (to each stop, the distance between its up\_tp and down\_tp)
- STOP\_TIME (ultimate goal of the database)
- TRIP\_NUMBER (will be very useful for future work)
- TP\_DISTANCE
- TIME\_POINT

So far the Stop table has the structure as following:

COL	ITEM NAME	WDTH	OPUT	TYP	N.DEC
1	VERSION_KEY	6	6	I	-
7	SERVICE_TYPE	1	1	C	-
8	RTE	3	3	C	-
11	DIR	1	1	C	-
12	PATH_NUMBER	6	6	I	-
18	LOC_ID	11	11	I	-
29	STOP_DISTANCE	16	16	N	10
45	UPDOWN_DIS	16	16	N	10
61	UPDOWN_TIME	11	11	I	-
72	STOP_TIME	11	11	I	-
83	TRIP_NUMBER	6	6	I	-
89	TP_DISTANCE	16	16	N	10
105	TIME_POINT	6	6	I	-

The result table has the same structure as the above structure. In the results table, information of the three items, SERVICE\_TYPE, TRIP\_NUMBER and STOP\_TIME, is the goal, which is got by the program computing. Other added items are mostly intermediate items, which are used in the process of computing.

### Description of the computing process

To each of the tables, the program will select a basic unit. The program operate action on the two basic units.

For the stop table, the program selects a subset according to three items: a RTE, a DIRECTION, a PATH\_NUMBER. After these three level queries, there is a single path of stops. There are 10 - 90s stops to each path, depending on the length of the path. The program will read this subset and compute the stop time. One complete path subset is following as an example: (Figure 1)

A complete path information from the Bus stop table  
(Figure 1, is about here)

To the Trip & Time\_point table, besides anchoring a RTE, a DIRECTION, a PATH\_NUMBER, two more field are needed to anchor a unique trip. These are: SERVICE\_TYPE and TRIP\_NUMBER. A unique trip has 5 - 10 time points. Here, the importance of adding the TRIP\_INCRE item is shown. Because, in the original Trip table, the item TRIP\_NUMBER may be the same between two different routes, or same routes different direction, or path. (Refer to Figure 2, as an example)

A complete trip information from the Trip & Time\_point table  
(Figure 2 is about here)

After decomposing the large tables into such a pair of small subsets. The program can now create relation between them and compute. To the above two subsets, the LOC\_ID is the only common item, by which the relation is to be established. The program will compute on these two subsets. Then the Stop\_time, Service\_type, and Trip\_number information is filed in each relevant items which has default value as 0 or NULL. Looping on the TRIP\_INCRE computes all such pairs of subsets. The result of the computation on each pair of subsets is appended to an ultimate result table.

The advantage of this process for pre-computing and saving the stop time lies in its high query speed for user. If the TTIA1 user query through the InterNet, It is a simple database query process. The disadvantage is that this result table needs a large space to save. It should not be difficult to maintain and revise the result table. If the dispatching system makes a new trip, the database maintainer needs to purge the old trip and run the program again just on the new trip and then append the new trip information. The alternative is to interpolate the schedule time at bus stop level on the fly.

The model which is used to interpolate stop\_time:

$Stop\_time = up\_tp\_time + (stop\_to\_up\_distance / up\_down\_distance) * up\_down\_time$

Where:

up\_tp\_time: The time to the upstream time\_point.

stop\_to\_up\_distance: the distance between the stop and its up\_time point.

up\_down\_distance: To a stop, this is zero when the stop is a time point too

Otherwise, it is the delta\_distance of its up and down TPs.

up\_down\_time: To a stop, this is zero when the stop is a time point too

Otherwise, it is the delta\_time of its up and down TPs.

This model is simplified, because it does not consider the speed variation on the network. However Tri-Met is placing pseudo time points into its system to improve the assumption of constant speed between time points. With the help of this method, the assumption of constant speed between two adjacent time points is acceptable.



### The result table

The result table is the stop level schedule. It includes all the information from the three bus tables, Stop, Time\_point, and Trip table. The following is the structure of the result table.

```
ENTER COMMAND >ITE
DATAFILE NAME: RESULT2                                09/03/1996
 13 ITEMS: STARTING IN POSITION  1
COL ITEM NAME      WIDTH OPUT TYP N.DEC
 1  VERSION_KEY    6      6      I  -
 7  SERVICE_TYPE   1      1      C  -
 8  RTE            3      3      C  -
11  DIR            1      1      C  -
12  PATH_NUMBER    6      6      I  -
18  LOC_ID         11     11     I  -
29  STOP_DISTANCE  16     16     N 10
45  UPDOWN_DIS     16     16     N 10
61  UPDOWN_TIME    11     11     I  -
72  STOP_TIME      11     11     I  -
83  TRIP_NUMBER    6      6      I  -
89  TP_DISTANCE    16     16     N 10
105 TIME_POINT     6      6      I  -
```

The STOP\_TIME, SERVICE\_TYPE, and TRIP\_NUMBER are those that this program produces. (Refer to Figure 3)

A small set of records from the result table  
Figure 3 is about here

### User entered data.

As a part of TTIA1, the user enters data from his or her computer. The data will be transferred to DBMS (in this project, INFO). Suppose the INFO database received the following data originally from the user:

- route
- weekday or weekend to take the bus
- time to take bus
- direction
- stop

After querying the database (specifically, the result table), the result is:  
the arrival time

This can easily be done in INFO just by the following commands: SELECT and RESELECT. The following is copied from INFO, which performs a query to the database:

```
ENTER COMMAND >SEL RESULT /* From the result table
45328 RECORD(S) SELECTED

ENTER COMMAND >RES RTE = '008' /* select route
19822 RECORD(S) SELECTED

ENTER COMMAND >RES SERVICE_TYPE = 'S' /* select service type
7070 RECORD(S) SELECTED

ENTER COMMAND >RES DIR = '0' /* select direction
3619 RECORD(S) SELECTED

ENTER COMMAND >RES LOC_ID = 6777 /* select stop location
63 RECORD(S) SELECTED

ENTER COMMAND >RES STOP_TIME > 40000 AND STOP_TIME < 45000
6 RECORD(S) SELECTED /* *select time range

ENTER COMMAND >DIS STOP_TIME
40,228
42,028
43,828
41,128
42,928
44,728 /* *a list of the arrival times
```

\*\* The time value is in the format of seconds post midnight.

In practice, a user can not know the LOC\_ID. The TTIA system does not expect a user can enter data such as LOC\_ID: 6777, or TIME 40000. The user will enter his or her data by clicking on a menu or clicking some points on a map, or by other means. How to let the user enter data is the question of the software interface. It will be a future work of TTIA.

## Conclusion

The database described in this paper was created on the INFO system. Using ARC/INFO to create such a large transit database system was an experiment. INFO, and other modules in ARC/INFO, e.g. ARC/PLOT and ARC, can do the DBMS operations.

However, they do not support standard SQL. Some large DBMS system, such as Oracle 7 may be more powerful. ARC/INFO has the strength in its functionality for mapping . In the future the project will need this functionality.

Tri-Met has done well in adjusting its database suitable for APTS systems. However, the contents of data files (or the tables) still have some problems. For example, in the Trip & Time table, when selecting the following time point: RTE = 014, DIR = 1, PATH\_NUMBER = 2, and TIME\_POINT = 1061, there are records. However, in the Time\_points table, there is no such time point. In other words, there is no information about its LOC\_ID or its TP\_DISTANCE. The program computes arrival time by TP\_DISTANCE.

The process above is a static process. The interpolate of the "exception record" is the next work of TTIA. To do this, the TRIP\_NUMBER will be helpful. The TRIP\_NUMBER can relate to TRAIN\_NUMBER. With the TRAIN\_NUMBER, the exception information may be included. This is a important task of the future work. With Loc-id, it is possible to display maps. With trip\_number, then relating to TRAIN\_NUMBER, it may be possible to receive exception reports. These will be future tasks.

## Reference

- Dueker, J, Kenneth and Vrana Ric. GIS Applications in Urban Public Transportation: A case Study of Tri-Met, :Portland, Oregon.
- Groff, Jonathan. Building Database Gateways With Inter-Application Communication (IAC)
- Peng, Zhongren and Dueker, J, Kenneth: Spatial Data Integration in Route\_level Transit Demand Modeling
- Peng, Zhongren and Dueker, J, Kenneth. An Enterprise GIS Database Design for Transit Applications.

Figure 1

(Internal index) VERSION\_KEY, SERVICE\_TYPE , RTE , DIR, PATH\_NUMBER, LOC\_ID, STOP\_DISTANCE, UPDOWN\_DIS, UPDOWN\_TIME, STOP\_TIME, TRIP\_NUMBER, TP\_DISTANCE, TIME\_POINT

1	00,024	033	1	4	8223	0.000000000	0.000000000	0	0	0	0.000000000	0
2	00,024	033	1	4	3672	426.883860000	0.000000000	0	0	0	0.000000000	0
3	00,024	033	1	4	3680	808.658860000	0.000000000	0	0	0	0.000000000	0
4	00,024	033	1	4	3657	2,670.605859999	0.000000000	0	0	0	0.000000000	0
5	00,024	033	1	4	3652	4,479.485269999	0.000000000	0	0	0	0.000000000	0
6	00,024	033	1	4	3797	5,612.534309999	0.000000000	0	0	0	0.000000000	0
7	00,024	033	1	4	3856	7,023.685399999	0.000000000	0	0	0	0.000000000	0
8	00,024	033	1	4	9418	7,260.393399999	0.000000000	0	0	0	0.000000000	0
9	00,024	033	1	4	3785	14751.227759999	0.000000000	0	0	0	0.000000000	0
10	00,024	033	1	4	3858	16209.462759999	0.000000000	0	0	0	0.000000000	0
11	00,024	033	1	4	3837	18656.264799999	0.000000000	0	0	0	0.000000000	0
12	00,024	033	1	4	2171	26208.917929999	0.000000000	0	0	0	0.000000000	0
13	00,024	033	1	4	2642	28173.812039999	0.000000000	0	0	0	0.000000000	0
14	00,024	033	1	4	5580	31525.768139999	0.000000000	0	0	0	0.000000000	0
15	00,024	033	1	4	5581	32099.163139999	0.000000000	0	0	0	0.000000000	0
16	00,024	033	1	4	7788	32914.658139999	0.000000000	0	0	0	0.000000000	0

Note:

The column SERVICE\_TYPE is defaulted as Null. The last several columns are defaulted as 0.

The program is to compute all these items and fill the computed values in result table.:

UPDOWN\_DIS,  
UPDOWN\_TIME,  
STOP\_TIME,  
TRIP\_NUMBER,  
TP\_DISTANCE,  
TIME\_POINT

Figure 2

(Internal index) RTE, DIR, PATH\_NUMBER, TIME\_POINT, TRIP\_NUMBER, TIME, SERVICE\_TYPE, LOC\_ID, TP\_DISTANCE, DELTA\_DIS, DELTA\_TIME, TRIP\_INCRE

1	033	1	3	3,301	1,070	74,220	U	8,758	0.000000000	0.000000000	0	167
2	033	1	3	335	1,070	74,460	U	136	8,576.837449999	8,576.837449999	240	167
3	033	1	3	334	1,070	74,760	U	3,791	16945.053419999	8,368.215969999	300	167
4	033	1	3	333	1,070	75,060	U	3,795	27077.706509999	10132.653090000	300	167
5	033	1	3	322	1,070	75,420	U	8,223	39014.989109999	11937.282600000	360	167
6	033	1	3	310	1,070	75,960	U	3,858	55526.677919999	16511.688810000	540	167
7	033	1	3	9,968	1,070	76,380	U	7,806	72201.867769999	16675.189850000	420	167
8	033	1	3	4,449	1,070	76,740	U	3,007	76790.371459999	4,588.503690000	360	167

Note:

This subset has the information of trip number, time, and time point, etc.

Figure 3

(Internal index) VERSION\_KEY, SERVICE\_TYPE , RTE , DIR, PATH\_NUMBER, LOC\_ID, STOP\_DISTANCE, UPDOWN\_DIS, UPDOWN\_TIME, STOP\_TIME, TRIP\_NUMBER, TP\_DISTANCE, TIME\_POINT

```
ENTER COMMAND >DIS
```

25507	00,024	W	008 0	00,001	0000007767	0.0000000000	7,241.9159499990	360	20,580	1,005	0.0000000000	9,968
25508	00,024	W	008 0	00,001	0000007800	529.0660000000	7,241.9159499990	360	20,606	1,005	0.0000000000	0
25509	00,024	W	008 0	00,001	0000007777	1,033.1115199999	7,241.9159499990	360	20,631	1,005	0.0000000000	0
25510	00,024	W	008 0	00,001	0000007803	1,548.5785599999	7,241.9159499990	360	20,656	1,005	0.0000000000	0
25511	00,024	W	008 0	00,001	0000007782	2,067.4174699999	7,241.9159499990	360	20,682	1,005	0.0000000000	0
25512	00,024	W	008 0	00,001	0000007751	2,584.5825299999	7,241.9159499990	360	20,708	1,005	0.0000000000	0
25513	00,024	W	008 0	00,001	0000007758	2,846.8220699999	7,241.9159499990	360	20,721	1,005	0.0000000000	0
25514	00,024	W	008 0	00,001	0000009298	3,356.2150699999	7,241.9159499990	360	20,746	1,005	0.0000000000	0
25515	00,024	W	008 0	00,001	0000001612	4,422.8491999999	7,241.9159499990	360	20,799	1,005	0.0000000000	0
25516	00,024	W	008 0	00,001	0000001097	7,228.2047399999	0.0000000000	0	20,940	1,005	7,241.9159499999	9,035
25517	00,024	W	008 0	00,001	0000009362	8,284.9331299999	3,118.2537999990	240	21,020	1,005	0.0000000000	0
25518	00,024	W	008 0	00,001	0000004042	8,828.2645399999	3,118.2537999990	240	21,062	1,005	0.0000000000	0
25519	00,024	W	008 0	00,001	0000004053	9,333.4733399999	3,118.2537999990	240	21,100	1,005	0.0000000000	0
25520	00,024	W	008 0	00,001	0000004055	9,843.7733399999	3,118.2537999990	240	21,140	1,005	0.0000000000	0
25521	00,024	W	008 0	00,001	0000009305	10,447.9187699990	0.0000000000	0	21,180	1,005	10360.1697499990	80
25522	00,024	W	008 0	00,001	0000004047	10,865.8657699990	5,306.7555400000	240	21,202	1,005	0.0000000000	0
25523	00,024	W	008 0	00,001	0000006842	11,686.5565599990	5,306.7555400000	240	21,239	1,005	0.0000000000	0
25524	00,024	W	008 0	00,001	0000009342	12,606.8868799990	5,306.7555400000	240	21,281	1,005	0.0000000000	0
25525	00,024	W	008 0	00,001	0000006843	13,125.3656599990	5,306.7555400000	240	21,305	1,005	0.0000000000	0
25526	00,024	W	008 0	00,001	0000006834	13,633.2196599990	5,306.7555400000	240	21,328	1,005	0.0000000000	0
25527	00,024	W	008 0	00,001	0000006815	14,548.5301899990	5,306.7555400000	240	21,369	1,005	0.0000000000	0
25528	00,024	W	008 0	00,001	0000006777	15,106.3541899990	5,306.7555400000	240	21,394	1,005	0.0000000000	0
25529	00,024	W	008 0	00,001	0000006799	15,666.9251899990	0.0000000000	0	21,420	1,005	15666.9252899990	83
25530	00,024	W	008 0	00,001	0000006811	16,244.5682999990	6,301.0201400000	240	21,442	1,005	0.0000000000	0
25531	00,024	W	008 0	00,001	0000006808	16,798.6572999990	6,301.0201400000	240	21,463	1,005	0.0000000000	0
25532	00,024	W	008 0	00,001	0000006797	17,358.2052999990	6,301.0201400000	240	21,484	1,005	0.0000000000	0
25533	00,024	W	008 0	00,001	0000006785	17,918.6412999990	6,301.0201400000	240	21,505	1,005	0.0000000000	0
25534	00,024	W	008 0	00,001	0000006775	18,389.8477299990	6,301.0201400000	240	21,523	1,005	0.0000000000	0
25535	00,024	W	008 0	00,001	0000006783	18,859.9587299990	6,301.0201400000	240	21,541	1,005	0.0000000000	0
25536	00,024	W	008 0	00,001	0000006806	19,250.8427299990	6,301.0201400000	240	21,556	1,005	0.0000000000	0
25537	00,024	W	008 0	00,001	0000006801	19,698.6807299990	6,301.0201400000	240	21,573	1,005	0.0000000000	0
25538	00,024	W	008 0	00,001	0000006810	20,149.6917299990	6,301.0201400000	240	21,590	1,005	0.0000000000	0
25539	00,024	W	008 0	00,001	0000006787	20,950.7831499990	6,301.0201400000	240	21,621	1,005	0.0000000000	0
25540	00,024	W	008 0	00,001	0000006819	21,451.5411499990	6,301.0201400000	240	21,640	1,005	0.0000000000	0
25541	00,024	W	008 0	00,001	0000006773	21,967.9451499990	0.0000000000	0	21,660	1,005	21967.9454299990	81
25542	00,024	W	008 0	00,001	0000006814	22,435.2014499990	6,671.8924000000	300	21,681	1,005	0.0000000000	0
25543	00,024	W	008 0	00,001	0000006781	22,895.7154499990	6,671.8924000000	300	21,701	1,005	0.0000000000	0
25544	00,024	W	008 0	00,001	0000006795	23,312.3384499990	6,671.8924000000	300	21,720	1,005	0.0000000000	0
25545	00,024	W	008 0	00,001	0000006792	24,111.8714499990	6,671.8924000000	300	21,756	1,005	0.0000000000	0
25546	00,024	W	008 0	00,001	0000006771	24,782.7854499990	6,671.8924000000	300	21,786	1,005	0.0000000000	0
25547	00,024	W	008 0	00,001	0000006790	25,574.7620699990	6,671.8924000000	300	21,822	1,005	0.0000000000	0
25548	00,024	W	008 0	00,001	0000006803	26,096.3755999990	6,671.8924000000	300	21,845	1,005	0.0000000000	0
25549	00,024	W	008 0	00,001	0000006780	26,832.4575999990	6,671.8924000000	300	21,878	1,005	0.0000000000	0
25550	00,024	W	008 0	00,001	0000001275	27,277.5036199990	6,671.8924000000	300	21,898	1,005	0.0000000000	0
25551	00,024	W	008 0	00,001	0000001267	28,639.8376199990	0.0000000000	0	21,960	1,005	28639.8378299990	85

Note:

The above is a small set of the result table. It has information from all the three other tables. It has the time value interpolated to bus stop level.

## Appendix 1: Algorithm

### Algorithm for main function

Purpose: The main routine selects subsets, joins tables, and adds items. It controls the loops. It reads and writes files.

Pre-requirement: The tables has been added the needed items

### Algorithm

Do loop, so all the trips will be included

- 1) Select subset which has information of a single trip from Trip\_Time table, by a trip index
- 2) Use "cursor first" in AML to get the items value of the first record.
- 3) Define variables to save the value of RTE, DIR, SERVICE TYPE, etc. of this trip
- 4) Select subset which has information of a single path from Bus Stop table, by RTE, DIR, PATH\_NUMBER.
- 5) Call sub routines to interpolate stop time
- 6) End loop

The algorithm above was coded in AML. The AML program was run in ARC/PLOT. In AML, there are some way to switch among several modules of ARC/INFO. For example, using directive "&data," the program change work platform among ARC, ARCPLOT and INFO. ARC/PLOT provide commands "Reselect," "Aselect" and "ClearSelect" for selecting subsets from tables. And the command "infile" is used to write the selected subset to a new data file.

### The algorithm for subroutines

Purpose: The subroutine computes the stop\_time to for selected subset. Also it writes the SERVICE\_TYPE and other intermediate items to the result table

ARC/INFO has a command to deal with each record of an INFO data file: "Cursor". Here, the CURSOR is not the meaning of the flickering dot on a monitor. It is a concept in INFO (In other SQL DBMS, the same command is "fetch"), which is used to access each record in the table.

- 1). Computing delta time and delta distance. (in Summary)
  - Read the subset which is from Trip & Time table
  - compute delta distance of two adjunct time points, add to tables SAD022
  - compute delta time of two adjunct time points, add to table SAD022
- 2). Computing the stop distance
  - Relate the two intermediate subset by LOC\_ID
  - Read the other subset which is from Bus Stop table
  - Compute up\_down TP distance of two adjunct time points.

Compute up down TP time of two adjunct time points.

Compute stop\_time.

Fill all the computed value to their relevant items which are default as 0 or Null

## Appendix 2

Because of the problem of no match Time\_point mentioned before in the conclusion. The program will have wrong result on some routes, e.g. 014. After Tri\_MET modify this problem, this program may run once to all the routes. Right now, the program run on route 008 and 017 as tests. Because these two routes do not have the above problem.

All the following steps can be written into AML MACRO. However, to do this must wait till the TRI\_MET modify its data files. The future work is write in a database server which supports SQL.

Guide for creating database in INFO

- \* Trip table: SADO22.FIN
- \* Time Point table: SADO23.FIN
- \* Bus Stop Table: SADO24.FIN

### 1. Update the tables structure of, and

After the update, the items are in the sequence of  
RTE, DIR, PATH\_NUMBER, TIMEPOINT

Do this in Arcplot:

```
reselect sad022.fin (or sad023.fin)
infofile sad022.fin sad022.fin ~
RTE DIR PATH_NUMBER TIME_POINT TRIP_NUMBER TIME ...
```

### 2. Redefine an item 'str' from 1 to 16 as Character.

Do this in INFO

### 3. Reselect the route 008 (then 017), which will be used to test the program.

Do this in Arcplot:

```
clearsel
resel SADO24.FIN info route = '008'
infofile SADO24.FIN INFO SADO24A.FIN
clearsel
resel SADO22.FIN info route = '008'
infofile SADO22.FIN INFO SADO22A.FIN
clearsel
```

For other routes, just change the '008'

### 4. Additems to SADO22A.FIN: LOC\_ID, TP\_DISTANCE

Do this in Arc

### 4. Additems in ARC.

The following items are added to SADO22A.FIN



DELTA\_DIS (which = tp distance - up tp distance)

DELTA\_TIME (which = tp time - up tp distance)

The following was added to the stop table (SAD024)

UPDOWN\_TIME (to each stop, the time between its up\_tp and down\_tp)

UPDOWN\_DIS (to each stop, the distance between its up\_tp and down\_tp)

STOP\_TIME (ultimate goal of the database)

TRIP\_NUMBER

TP\_DISTANCE

TIME\_POINT

5. Join table SAD023A.FIN to SAD022A.FIN

Do this in INFO

```
SEL SAD022.FIN
```

```
REL SAD023.FIN BY STR
```

```
CAL LOC_ID = $1LOC_ID
```

```
CAL TP_DISTANCE = $1TP_DISTANCE
```

6. Add an items named TRIP\_INCRE for unique trip\_number to SAD022A.FIN (Which is now have both Trip and Time\_point information). Save as TRY\_CURSOR.

Do this in ARC

Then run the program *add\_incre.aml* in ARC/PLOT to fill the contents in

7. Get the TRIP\_INCRE maximum value by observing the data file. This can be do dynamically by a subroutine. Use "fetch last" in SQL can do thia. In INFO, first sort on TRIP\_INCRE descendt, then use "cursor first".

8. Run the program *try\_main2.aml* in ARC/PLOT.

Some indices for controlling loops change according to different route: the index for path\_number, and trip\_number. For each route, these indices in the main function need to be changed, the subroutines keep unchanging.

9. Do these steps again on route 017. The file name changes. The value of control flow in main porgram need to be changed.